UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

PROYECTO DE GRADO

"SISTEMA DE INFORMACIÓN DE AUTOEVALUACIÓN DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES"

(CASO: UNIDADES ACADÉMICAS PREGRADO)

Proyecto de Grado para obtener el Título de Licenciatura en Informática
Mención: Ingeniería de Sistemas Informáticos

POR: RAQUEL GLADYS YUJRA CHOQUETOPA
TUTORA METODOLÓGICA: M.SC. ROSA FLORES MORALES
ASESORA: LIC. MENFY MORALES RIOS

LA PAZ – BOLIVIA
Julio, 2018

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

DEDICATORIA

A mis papá Teodoro Yujra y mi mamá Gregoria que con todo su amor y dedicación me han apoyado y guiarme, nunca tendré con que agradecerte todo lo que hacen por mí y mi hija.

A mi hija Shecid y esposo Israel que son el pilar de mi vida, y el motivo de superación constante.

A toda mi familia por la confianza brindada todo este tiempo.

Raquel Gladys Yujra Choquetopa

AGRADECIMIENTOS

A mis asesora la M.Sc.. Rosa Flores Morales quien me apoyo con su conocimiento paso a paso para la realización de la presente proyecto, con su pedagogía y animo de enseñar.

A mi tutor de tesis la Lic. Menfy Morales Rios quien me fue corrigiendo, orientando e impulsando en la realización del presente proyecto, dedicándome su tiempo para poder brindar sus consejos y sabiduría con su larga experiencia y conocimiento en el campo de la investigación.

A la mi M.Sc. Elizabeth García Escalante quien me permitió desarrollar el presente proyecto, guiándome y brindándome el apoyo e información necesaria con su gran experiencia.

A mis amigos quienes compartieron momentos alegres conmigo durante la vida universitaria e hicieron más divertida e interesante esta etapa de mi vida.
Gracias por cada lección de vida y prueba puesta en mi camino, que haríamos sin Dios en nuestras vidas, gracias a Dios por brindarme fortaleza y perseverancia para alcanzar mis propósitos y metas.

¡Muchas gracias!!!

RESUMEN

El sistema de información de autoevaluación es un proyecto, para ejecutar el proceso de autoevaluación de manera eficiente y efectiva basado en un modelo teórico de gestión para la acreditación de las carreras de pregrado de la Facultad de Ciencias Puras y Naturales.

Ya que en la actualidad, las universidades, están asumiendo con mucha seriedad los procesos de autoevaluación institucional. Como un mecanismo de respuesta a las exigencias de la gestión de calidad y excelencia.

El sistema cuenta con un módulo donde se inicia el proceso de autoevaluación, el cual la comisión encargada de ese proceso pueda preparar los formularios según a los actores que intervengan, es la asignación de componentes, por otro lado se cuenta con el módulo de la publicación del formulario al usuario y el respectivo control de los resultados obtenidos, todo esto facilita el proceso de autoevaluación ya que los cuestionarios se llenaran virtualmente y los resultados obtenidos en tiempo real.

Para el desarrollo de sistema se usa la metodología Scrum combinándolo con la metodología UWE para el análisis y diseño en base a los requerimientos recolectados de los procesos de autoevaluación.

Para determinar la calidad del sistema, se emplea la norma ISO 9126 con todas las características necesarias y también cuenta con todas las medidas de seguridad para el buen funcionamiento y custodia de la información.

Palabras clave:

autoevaluación, sistema de autoevaluación, ingeniería de software, scrum, gestión de calidad.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPITULO I
MARCO INTRODUCTORIO

1.1. INTRODUCCIÓN

La presencia de las diversas instituciones ofreciendo servicios a su entorno mediante aplicaciones web va creciendo exponencialmente, esto debido a la comunicación constante que se debe de tener con el cliente, y a la necesidad de consultar información en cualquier momento.

Para iniciar un proceso de autoevaluación, la Facultad Ciencias Puras y Naturales deben estar preparadas tecnológicamente para poder generar servicios web de acuerdo a la demanda de sus diferentes unidades académicas y hacen que este proceso sea periódico, accesible y de mejor costo.

En anteriores procesos de autoevaluación que se realizó en algunas carreras de la Facultad, se pudo evidenciar que, si bien el proceso de autoevaluación estaba sistematizado, los formularios eran llenados físicamente y el procesamiento de datos y elaboración de resultados para el informe final fueron llenados en hojas Excel.

El objetivo de este proyecto es el de agilizar los procesos anteriormente mencionados en las distintas unidades, de igual forma reducir considerablemente la cantidad de documentación física generada, disminución el tiempo en cada uno de los procesos y estar preparados para el crecimiento de sus servicios informáticos.

En este sentido el presente trabajo se propuso desarrollar un "SISTEMA DE INFORMACIÓN DE AUTOEVALUACIÓN DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES" en el caso de unidades académicas de pregrado,

Este sistema permitirá automatizar los procesos de autoevaluación en su totalidad y llegar a un sistema escalable y predispuesto a la integración con otros sistemas.

1.2. PLANTEAMIENTO DEL PROBLEMA

1.2.1. PROBLEMA GENERAL

De acuerdo al Reglamento General de Evaluación y Acreditación de Carreras o Programas de la Universidad Boliviana aprobado el XI Congreso Nacional de Universidades (Tarija2009) se entiende por autoevaluación. "Al proceso de recolección de información que, analizada e interpretada a la luz del marco referencial, posibilita la emisión de juicios de valor sobre las condiciones de funcionamiento de la carrera, dar cuenta de la calidad y pertinencia de la misma y conduce a la toma de decisiones relacionadas a la acreditación o postergación de la misma". (Ugarte, 2009)

En la Facultada de Ciencias Puras y Naturales cuenta con seis carreras de las cuales la Carrera de Informática realizo su autoevaluación, hecho que permitió reconocer a la carrera como una Unidad Académica Acreditada, por un periodo entre el 11 de diciembre del 2013 al 11 de diciembre del 2019, la Carrera de Estadística inició el proceso de autoevaluación y acreditación a fines del año 2010 y concluyó el 2014 y la Carrera de Matemática realizo su autoevaluación el año 2000, de tal forma que las otras carreras restantes están aún en un plan de iniciar un proceso de autoevaluación.

La autoevaluación en las unidades, permiten consolidar la cultura de la Autoevaluación y Autorregulación Institucional, fortaleciendo la credibilidad y el reconocimiento por parte de la comunidad universitaria y la sociedad, en el marco normativo y regulatorio de nuestra Casa Superior de Estudios.

La recolección de información por parte de los encargados del Departamento de Evaluación, Acreditación y Gestión de la Calidad. Se puede asegurar que el desarrollo de este proceso a los diferentes actores como la HCC, docentes, estudiantes, administrativos e investigadores, de las carreas de la UMSA, están sistematizados y a disposición de los interesados, sin embargo, no son utilizados, por diversas razones una de ellas puede ser que no se conozca la funcionalidad o que el sistema sea complejo para entender, por la misma situación en muchos casos la aplicación de los formularios de autoevaluación son llenados de forma física y manual, y posteriormente se toma el tiempo en transcribir la información, para conocer los resultados y la generar graficas de información, se las realizan en el paquete de Excel.

Todo este sumario de llenado de formularios y obtención de resultados de forma física lleva a la acumulación de papel que posteriormente es innecesario, la tarea de transcribir genera la duplicidad de esfuerzos y la pérdida de tiempo valioso en estos procesos manuales con el riesgo de presentar errores.

1.2.2. PROBLEMAS SECUNDARIOS

- Llenado manual de diversos formularios oficiales, lo que ocasiona errores de transcripción "errores humanos" y demora.

- Acumulación de gran cantidad de documentación física que posteriormente no son útiles para la obtención de información.

- Omisión o mala interpretación de normas y/o ponderaciones establecidas para dicho proceso por el Reglamento de Autoevaluación.

- Retraso de resultados finales por cantidad de documentación a transcribir.

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

Analizar, diseñar e implementar un sistema de información web para el proceso de autoevaluación en las carreras de la Facultad de Ciencias Puras y Naturales que ayuden a simplificar todas las tareas.

1.3.2. OBJETIVO ESPECÍFICO

- Recopilara información para los formularios de autoevaluación y los procesos de resultados del Departamento de Evaluación, Acreditación y Gestión de la Calidad.
- Analizar, diseñar una plataforma web empleando la metodología SCRUM, utilizando herramientas de software libre y certificados.
- Generar resultados del proceso de Autoevaluación respecto al marco de referencias del Sistema de la Universidad Boliviana
- Emitir reportes finales y medios para la correcta toma de decisiones sobre los procesos de autoevaluación.

1.4. JUSTIFICACIÓN

1.4.1. JUSTIFICACIÓN ECONOMICA

Con el nuevo sistema de administración los procesos se reducirán en relación al tiempo y los costos que se involucran en cuanto al uso del material de escritorio así mismo los esfuerzos humanos.

En relación a los costos por el desarrollo e implementación del nuevo sistema, estos serán mínimos, puesto que la propuesta hecha anteriormente se enfoca a trabajar con herramientas de software libres y certificadas, de igual forma se reducirán los costos para su mantenimiento

1.4.2. JUSTIFICACIÓN SOCIAL

El proyecto beneficiará a la comunidad universitaria ya que tendrán a mano las autoevaluaciones y las puntuaciones de los mismos, se podrá realizar en cualquier lugar, en el tiempo asignado.

Así mismo será mayormente beneficiada la población estudiantil ya que estará constantemente informada del estado y resultados de los procesos de autoevaluación.

JUSTIFICACIÓN TECNOLÓGICA

La Facultad de Ciencias Puras y Naturales cuenta con una conexión estable y eficiente de internet en todos sus ambientes, para poder conectarse al sistema, en el desarrollo del sistema se utilizará para los servicios del *Back-end* PHP que permite una comunicación sin pérdida de velocidad sin importar la cantidad de usuarios que estén conectados por su procesamiento por hilos, el *Front-end* será desarrollado en AngularJS por su velocidad, así mismo se trabajará con el gestor de base de datos *MySQL*.

1.5. ANTECEDENTES DE PROYECTOS SIMILARES

Existen diversos proyectos realizados que pueden ser usados como referentes para la administración de procesos universitarios, sin embargo, cada uno presenta características particulares con una aplicación específica debida a los requerimientos que cada institución presenta.

Proyectos que fueron elaborados en la Carrera de Informática de la Universidad Mayor de San Andrés:

- Sistema Web Integrado de Gestión Académica Administrativa Caso: C.E.C.O.M.P.
  Diego Paredes Mendoza, 2015. UNIVERSIDAD MAYOR DE SAN ANDRÉS, FACULTAD DE CIENCIAS PURAS Y NATURALES.
  Con el propósito de realizar un sistema bajo la plataforma web que permite gestionar el área administrativa y académica del instituto CECOMP. Utilizando SCRUMBAN como metodología de desarrollo y UWE como metodología de modelado.

- Sistema de Control y Generación de Partes de Asistencia de Docentes y Carga Horaria Académica Caso: Facultad de Ciencias Puras y Naturales
  Sergio Adrián Flores Gutierrez, 2016, UNIVERSIDAD MAYOR DE SAN ANDRÉS, FACULTAD DE CIENCIAS PURAS Y NATURALES
  Con el propósito de realizar un sistema web que permite el control de partes de asistencia de los docentes de la Facultad de Ciencias Puras y Naturales. utilizando SCRUM como metodología de desarrollo y WebML como metodología de modelado, desarrollado mediante el framework Laravel 5.0, con el gestor de base de datos MySQL.

- Sistema Web de Control y Seguimiento de Documentación Caso: Departamento de Infraestructura de la U.M.S.A.
  Wilmer Mijhael Yucra Lecoña, 2016, UNIVERSIDAD MAYOR DE SAN ANDRÉS, FACULTAD DE CIENCIAS PURAS Y NATURALES.
  Con el propósito de realizar un sistema web que permite el control y seguimiento de la documentación del Departamento de Infraestructura de la U.M.S.A., Utilizando

como metodología de desarrollo SCRUM y para el modelado UWE, desarrollado con el *framework CodeIgniter* 2 y el gestor de base de datos MySQL.

1.6. ALCANCES Y LÍMITES

1.6.1. ALCANCES

Los alcances del proyecto de administración para la autoevaluación en las unidades académicas son:

- Módulo del inicio de administración de usuarios docentes, estudiantes, administrativos y autoridades que se encargará del registro correcto de los usuarios al sistema.

- Módulo de selección, que permitirá la selección de preguntas factibles para la construcción del formulario de una unidad

- Módulo de aplicación a la encuesta, que permitirá al usuario autoevaluarse, llenando el formulario propuesto por la comisión autorizada.

- Módulo de reportes, que se encargará de generar reportes necesarios para la obtención de resultados de los formularios llenados.

1.6.2. LÍMITES

Los límites del presente proyecto según un análisis de requerimientos en relación a la autoevaluación son:

- No se realizará el módulo de acreditación.

- En la autoevaluación se requieren mediciones y cuantificaciones de la infraestructura y de los servicios, esta información debe ser registrada manualmente.

CAPÍTULO II
MARCO TEÓRICO

2.1. INTODUCCIÓN

En este capítulo se describe y se brinda los conocimientos de los principios y conceptos básicos para la realización del proyecto, sin embargo no se puede dar una teoría completa y extensa de las metodologías, técnicas y herramientas que se utilizara, por el contrario el objetivo del mismo es de presentar una base para la fácil comprensión del proyecto.

2.2. INGENIERÍA DE SOFTWARE

Ingeniería de Software La ingeniería de Software es una disciplina o área de la información que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelven temas de todo tipo. Esta definición permite incluir dentro de la ingeniería de software a un gran número de áreas en la informática, desde desarrollos de sistemas operativos, construcción de compiladores, hasta los nuevos desarrollos de software para aplicaciones basadas en aplicaciones enriquecidas en la web (Pressman, 2012).

La ingeniería de Software es una tecnología con varias capas (Herramientas, Métodos, Procesos, entre otros), cualquier enfoque de ingeniería debe basarse en un compromiso organizacional de calidad.

El proceso define una estructura que debe establecerse para la obtención eficaz de tecnologías de ingeniería de software.

El proceso de software forma la base para el control de la administración de proyectos de software, y establece un contexto en el que se aplican métodos técnicos, se generan modelos de trabajo, se establecen puntos de referencia, se asegura la calidad de software y se adapta a cambios de manera apropiada.

Los métodos de la ingeniería de software proporcionan la experiencia técnica necesaria para elaborar software, que incluye un conjunto amplio de tareas, como comunicación, análisis de los requerimientos, modelación del diseño, construcción del programa, pruebas y apoyo. Las herramientas de la ingeniería de software proporcionan un apoyo automatizado o semiautomatizado para el proceso y los métodos. Cuando se integran las herramientas de modo que la información obtenida pueda ser utilizada por otros actores, queda establecido un sistema llamado ingeniería de software asistido por computadora que apoya el desarrollo de software. (Vliet & Van, 2002)

A continuación se detallan las herramientas más importantes:

- Requerimientos de Software. Un requerimiento se define como una exigencia que debe ser cumplida para dar solución a un problema específico. Existen diferentes áreas de análisis como ser la especificación de requerimientos, análisis, validación, clasificación, negociación, etc.
- Diseño del Software. Es el proceso en el cual se define la arquitectura, componentes, interfaces y otras características relativas al desarrollo del software mismo.
- Construcción del Software. Referida a la creación en detalle del Software a través de la combinación de las diferentes herramientas de codificación, verificación, pruebas de unidad, pruebas de integración y depuración.
- Pruebas del Software. Proceso de verificación dinámica del comportamiento del Software ante un conjunto limitado de casos de prueba.
- Mantenimiento del Software. Estas actividades de mantenimiento comienzan generalmente cuando el Software entra en la etapa de pruebas de funcionalidad con datos reales, pruebas que son realizadas por parte de los usuarios finales con la finalidad de que estos puedan adaptarse a los cambios, así mismo puedan sugerir cambios para su posterior adaptabilidad.

2.2.1. ROLES DE SCRUM

Los roles centrales son aquellos que su participación es indispensable para la realización del proyecto, están comprometidos con el proyecto y son responsables del éxito de cada *sprint* y del proyecto en general. Estos son:

- Propietario del producto (*Product Owner*): El propietario del producto es quien toma las decisiones del cliente. Su responsabilidad es el valor del producto. Decide en última instancia cómo será el resultado final, y el orden en el que se van construyendo los sucesivos incrementos (que se pone y que se quita de la pila del producto), y cuál es la prioridad de las funcionalidades. Conoce el plan del producto, sus posibilidades y plan de inversión, así como del retorno esperado a la inversión realizada, y se responsabiliza sobre fechas y funcionalidades de las diferentes versiones del mismo. Para ejercer este rol es necesario (Palacio, 2014)

    - Conocer perfectamente el entorno del negocio del cliente, las necesidades y el objetivo que se persigue con el sistema que se está construyendo.

    - Tener la visión del producto, así como las necesidades concretas del proyecto, para poder priorizar eficientemente el trabajo.

    - Recibir y analizar de forma continua retroinformación del entorno de negocio (evolución del mercado, competencia, alternativas) y del proyecto (sugerencias del equipo, alternativas técnicas, pruebas y evaluación de cada incremento).

- Scrum Master: Según Palacio el Scrum Master es el responsable del cumplimiento de las reglas del marco técnico Scrum, asegurándose de que se entiendan en la organización, y se trabaje conforme a ellas. El Scrum Master realiza su trabajo con un modelo de liderazgo servil, al servicio y en ayuda del equipo y del propietario del producto, proporcionando:

    - Asesoría y formación al equipo para trabajar de forma auto organizada y con responsabilidad de equipo.
    - Revisión y validación de la pila del producto.
    - Moderación de las reuniones.
    - Resolución de impedimentos que en el sprint puedan entorpecer la ejecución de las tareas.
    - Gestión de las dinámicas del grupo en el equipo.
    - Configuración, diseño y mejora continua de las prácticas de Scrum en la organización, respeto de la organización y los implicados, con las pautas de tiempos y formas de Scrum.

- Equipo de Desarrollo: Según Palacio (2014) el Equipo de Desarrollo, lo forman el grupo de profesionales que realizan el incremento de cada sprint. Se recomienda que un equipo Scrum tenga entre 4 a 8 personas. No son considerados parte del grupo de desarrollo al Scrum Master ni al Propietario del producto. Cada uno tiene tareas asignadas y responsabilidades, siguiendo un proceso de ejecución. Es posible que algunos miembros sean especialistas en áreas concretas, pero la responsabilidad recae sobre el equipo de desarrollo en conjunto. Entonces en el equipo:

- Todos conocen y comprenden la visión del propietario del producto.
- Aportan y colaboran con el propietario del producto en el desarrollo de la pila del producto.
- Comparten de forma conjunta el objetivo de cada sprint y la responsabilidad del logro.
- Todos los miembros participan en las decisiones.
- Se respetan las opiniones y aportes de todos.

En la Figura 2.1 se puede ver un gráfico como resumen de los roles expuestos:

Figura 2. 1. Descripción resumida de los Roles en Scrum
Fuente: (Palacio, 2014)

2.2.2. ARTEFACTOS

- Pila del producto: (*Product backlog*): La pila del producto es el inventario de funcionalidades (Historias de Usuario), mejoras y corrección de errores que deben incorporarse al producto a través de los sucesivos *sprints*. Representa todo aquello que esperan el cliente, los usuarios, y en general los interesados. Todo lo que suponga un trabajo que debe realizar el equipo debe estar reflejado en esta pila. Sin embargo, la pila de requisitos del producto nunca se da por completada; está en continuo crecimiento y evolución. Al comenzar el proyecto incluye los requisitos inicialmente conocidos (generalmente se empieza con una tormenta de ideas o un proceso de exploración donde todo el equipo de desarrollo colabora en base a la visión del propietario del producto) y mejor entendidos, y conforme avanza el desarrollo, y evoluciona el entorno en el que será usado, se va desarrollando (Palacio, 2014).

  Los elementos de la pila del producto que pueden ser incorporados a un sprint se denominan "preparados" o "accionables" y son los que pueden seleccionarse en la reunión de planificación del sprint. La pila del producto no es un documento de requisitos, sino una herramienta de referencia para el equipo. Es recomendable que al menos tenga la siguiente información:

  - Identificador Único de la funcionalidad o trabajo.
  - Descripción de la funcionalidad/requisito, denominado "historia de usuario".
  - Campo o sistema de Priorización.
  - Estimación del esfuerzo necesario.

Dependiendo del proyecto, y organización del equipo, pueden ser aconsejables otros campos más:

- Observaciones
- Persona Asignada
- Nº de Sprint en el que se realiza
- Modulo del sistema al que pertenece

En la Tabla 2.1 podemos ver un ejemplo de una pila de producto con los campos esenciales:

Tabla 2. 1. Ejemplo de una Pila de Producto
Fuente: (Palacio, 2014)

D	Prioridad	Descripción	Estimación
1	Muy Alta	Plataforma Tecnológica	30
2	Muy Alta	Interfaz de Usuario	40
3	Muy Alta	Un Usuario se registra en el sistema	40
4	Alta	El operador define el flujo y textos de un expediente	60
5	Alta		999

Pila de sprint (*Sprint backlog*): La pila de sprint es la lista que descompone las funcionalidades de la pila del producto (historias de usuario) en tareas necesarias para construir un incremento: una parte completa y operativa del producto.

La realiza el equipo durante la reunión de planificación del sprint, auto asignando cada tarea a un miembro del equipo, e indicando en la misma lista cuanto tiempo o esfuerzo se prevé que falta para terminarla.

La pila del sprint descompone el trabajo en unidades de tamaño adecuado para monitorizar el avance a diario, e identificar riesgos y problemas sin necesidad de procesos de gestión complejos.

Durante el sprint, el equipo actualiza a diario en ella los tiempos pendientes de cada tarea. Al mismo tiempo, con estos datos se traza el grafico de avance o trabajo consumido (Palacio, 2014).

- Incremento: El incremento es la parte del producto producida en un sprint, y tiene como característica el estar completamente terminada y operativa, es decir, en condiciones de ser entregada al cliente.

  No se deben considerar como Incremento a propósitos, módulos o sub-módulos, ni partes pendientes de pruebas o integración. Sin embargo, es una excepción frecuente el primer sprint. En el que objetivos del tipo "contrastar la plataforma y el diseño" pueden resultar necesarios, e implican trabajos de diseño o desarrollo de prototipos para contrastar las expectativas de la plataforma o tecnología que se va a emplear (Palacio, 2014).

- Gráfico de Avance: También se suele llamar a este gráfico con su nombre inglés: "*burn-down*". Lo actualiza el equipo en el Scrum diario, para comprobar el ritmo de avance, y detectar desde el primer momento si es el previsto, o por el contrario se puede ver comprometida o adelantada la entrega prevista al final del sprint. La estrategia ágil para el seguimiento del proyecto se basa (Palacio, 2014).

  - Medir el trabajo que falta, no el realizado.
  - Seguimiento cercano del avance (diario de ser posible).

El equipo dispone en la pila del sprint, de lista de tareas que va a realizar, y en cada una se muestra el esfuerzo pendiente. Esto es, el primer día, en la pila de tareas se puede ver para cada tarea el esfuerzo que se ha estimado, puesto que aún no se ha trabajado en ninguna de ellas. Día tras día, cada miembro del equipo actualiza en la pila de sprint el tiempo que le queda a las tareas que va desarrollando, hasta que se terminan, es decir hasta que el tiempo pendiente sea cero (Palacio, 2014).

A continuación podemos ver un resumen grafico de los artefactos que se usan en la metodología Scrum.

Figura 2. 2. Descripción resumida de los Artefactos Scrum
Fuente: (Palacio, 2014)

2.2.3. FLUJO DE TRABAJO

- Planificación del Proyecto: También es conocido como la preparación del proyecto, que corresponde a la definición de las historias de usuario para la pila del producto

en base a los requerimientos y necesidades del cliente, y la determinación del tamaño y la duración de los sprints y cuántos de estos se realizarán, además conjuntamente también se realizan las actividades preparatorias de análisis, diseño, selección de las herramientas tecnológicas, identificación de políticas de calidad y seguridad para el sistema a desarrollar, y si es necesario la capacitación del equipo de desarrollo. Deben estar presentes el Propietario del Producto, el Scrum Master, el equipo de desarrollo, los interesados y dura de una a dos semanas (Menzinsky, 2015).

- Reunión de planificación de Sprint: En esta reunión se toman como base las prioridades y necesidades de negocio del cliente, y se determinan cuáles y cómo van a ser las funcionalidades que se incorporarán al producto en el siguiente sprint (Palacio, 2014).

El responsable de que se realice esta reunión es el Scrum Master y debe asegurarse de que estén presentes el propietario del producto, el equipo de desarrollo y si es necesario los implicados o interesados del proyecto. La reunión de planificación de Sprint dura aproximadamente una jornada de trabajo completa, cuando se trata de planificar un sprint largo (de un mes de duración) o un tiempo proporcional para planificar un sprint más breve (Palacio, 2014).

El propietario del producto presenta la pila de producto, exponiendo los requisitos de mayor prioridad que se necesita y prevé que se podrán desarrollar en el siguiente sprint. Si algún elemento de la pila del producto ha tenido cambios significativos desde la anterior reunión, explica las causas que los han ocasionado (Palacio, 2014).

El equipo de desarrollo hace preguntas y pide aclaraciones necesarias, además pueden proponer sugerencias, modificaciones y soluciones alternativas, como respuesta el

propietario del producto atiende las dudas y sugerencias del equipo asegurándose de que tengan un buen entendimiento de la idea y visión del producto.

El equipo desglosa cada funcionalidad en tareas, y estiman el tiempo para cada una de ellas, auto asignándose en base a sus conocimientos, intereses y una distribución homogénea de trabajo, componiendo así las tareas que forman la pila de (Palacio, 2014).

- Sprint: Se denomina sprint a cada ciclo o iteración de trabajo que produce una parte del producto terminada y funcionalmente operativa (incremento). La duración de cada sprint puede ser desde una hasta seis semanas, aunque se recomienda que no excedan de un mes. El equipo monitoriza la evolución de cada sprint en reuniones breves diarias (reuniones de pie – stand-up meeting) que duran entre 5 a 15 minutos donde se revisa en conjunto las tareas realizadas por cada miembro el día anterior, y previsto para el día en curso (Palacio, 2014).

- Reunión Diaria: Las reuniones diarias, no debe durar más de 15 minutos, deben estar presentes todo el equipo. En esta reunión cada miembro del equipo explica lo que ha logrado desde la anterior reunión diaria, lo que va a hacer en la jornada laboral actual y si está teniendo algún problema, o si prevé que puede encontrar algún impedimento, para que los demás miembros del equipo puedan colaborarle en solucionarlo (Palacio, 2014).

- Revisión del sprint: Se lo realiza al finalizar un sprint para comprobar el incremento. No debe durar más de 4 horas en caso de sprints largos. Para sprints de 1 o 3 semanas, de 1 a 2 horas es suficiente y deben estar presentes el equipo de desarrollo, el propietario del producto y el Scrum Master. El equipo expone el objetivo del sprint, las listas de funcionalidades deseadas, y las que se han desarrollado, haciendo una demostración de las partes construidas. Luego se hace

una ronda de preguntas y sugerencias, donde se genera información valiosa respecto a la visión del producto. El propietario del producto comprobando el progreso del sistema, ve si se puede considerar si se ha logrado el objetivo y el Scrum Master, de acuerdo a las agendas del propietario del producto y el equipo, cierra la reunión para la preparación del siguiente sprint (Palacio, 2014).

- Retrospectiva del sprint: Se los realiza tras la revisión de cada sprint, y antes de la reunión de planificación del siguiente sprint, puede durar de 1 a 3 horas, según la duración del sprint terminado. En ella el equipo realiza un autoanálisis de su forma de trabajar, identificando fortalezas para consolidarlos y puntos débiles para tomar acciones de mejoras (Palacio, 2014).

En la siguiente figura se ve un resumen gráfico de los eventos Scrum que ya fueron expuestos.

Figura 2. 3. Descripción Resumida de los Eventos Scrum
Fuente: (Palacio, 2014)

Por último, en la Figura 2.4 se puede ver un resumen del flujo de trabajo del marco técnico Scrum.

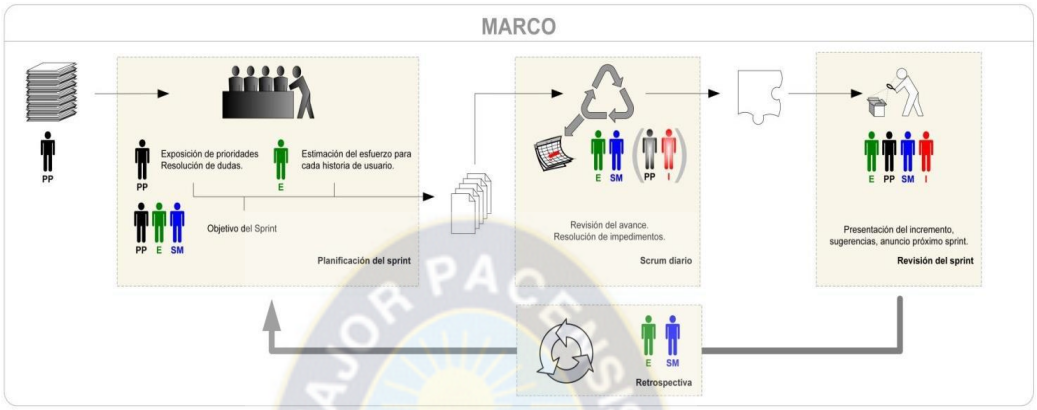


Figura 2. 4. Flujo de trabajo del Marco Técnico Scrum
Fuente: (Palacio, 2014)

2.3. MODELO DE PROCESO

Scrum es un método de desarrollo muy simple, que requiere mayor trabajo ya que no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. El ciclo de vida de Scrum está compuesto por tres tareas o etapas, *Pre-Game, Game y Post-Game*.

2.3.1. PRE – GAME

Antes de llevar a cabo el desarrollo del proyecto, se especifica lo que se va a realizar en las iteraciones, además de la prioridad con la que se lo hará, ésta fase consta de dos puntos importantes:

- Planeación. Durante la planeación todos los miembros del equipo incluyendo el cliente contribuyen a la creación de una lista de características del software, para el análisis y conceptualización del problema

Las tareas que se realizan en esta primera etapa son:

  - Recopilación de requerimientos para conformar el backlog del producto, priorizados de acuerdo a una evaluación del cliente.
  - Definición de las fechas de entrega de los *sprints* y sus funcionalidades.
  - Análisis de riesgos y controles apropiados para los riesgos.
  - Selección de las herramientas y de la infraestructura de desarrollo.
  - Cálculo y estimación del costo de cada iteración.

- Arquitectura. El objetivo de esta etapa es diseñar el cómo los elementos del backlog del producto serán puestos en ejecución. Esta fase incluye una revisión de la arquitectura del software y diseño de alto nivel.

En esta etapa se realizan las tareas de:

  - Revisión de los ítems del *backlog* del producto
  - Análisis de dominio para reflejar el nuevo contexto y requisitos del sistema.
  - Revisión de la arquitectura del software de acuerdo a los requisitos definidos.
  - Revisión del diseño de alto nivel.

2.3.2. GAME

Una vez realizada la especificación correspondiente, se lleva a cabo la elaboración del proyecto con un continuo seguimiento a cargo del mismo grupo de desarrollo.

En cada iteración del *Game* se realizan las siguientes tareas:

- Planeación del Sprint. Antes de comenzar cada Sprint o Iteración, se lleva a cabo dos reuniones consecutivas, en la primera se mejora y se prioriza nuevamente el backlog del producto, además de elegir las metas de la iteración. En la segunda reunión se deben considerar el cómo alcanzar los requerimientos y crear el backlog del Sprint. (Alaimo, 2013)

- Desarrollo del Sprint. El trabajo generalmente se organiza en iteraciones de 30 días (*Sprints*). El Sprint es el desarrollo de la nueva funcionalidad para el producto. Esta fase provee la siguiente documentación. Backlog del Sprint con las actividades desarrolladas, los responsables y la duración de cada actividad. (Alaimo, 2013).

- Revisión del Sprint. Al final de cada iteración se lleva a cabo una reunión de revisión, donde se presenta la nueva funcionalidad del software, las metas e inclusive la información de las funciones, diseño, ventajas, inconvenientes y esfuerzos del equipo. (Alaimo, 2013).

2.3.3. POST – GAME

Luego de haber culminado todas las iteraciones, solo resta la revisión final, denominada según la metodología Scrum, el cierre.

- Cierre. En ésta última etapa se realiza la preparación operacional, incluyendo la documentación final necesaria para la presentación. También en esta etapa se debe realizar dependiendo del tipo del producto, ya sea el entretenimiento del usuario o el marketing para la venta del nuevo software. (Alaimo, 2013).

2.4. INGENIERÍA WEB

La ingeniería Web es el proceso utilizado para crear, implantar y mantener aplicaciones y sistemas Web de alta calidad.

El proceso de Ingeniería Web está relacionado con la inmediatez y evolución, estos crecen continuamente para conllevar un proceso incremental y evolutivo, que permite al usuario involucrarse activamente, facilitando el desarrollo de productos que se ajustan a lo que se está buscando en cuanto a la funcionalidad del producto.
El proceso de desarrollo de aplicaciones hipermedia con lleva a la realización de una serie de actividades que son aplicables a cualquier Sistema Web, independientemente del tamaño y complejidad del mismo, la Ingeniería Web aplica el desarrollo, operación y mantenimiento de todo el ciclo de vida del desarrollo de Software.

Es importante mencionar que el impacto de los sistemas y aplicaciones basados en metodologías Web es el suceso más significativo en la era de las Tecnologías de Información y Comunicación. Conforme a la importancia de las Aplicaciones Web que crece, surge la necesidad de enfocar la ingeniería Web a un entorno disciplinado y bien estructurado.

Visualizando la evolución para las personas que están en el área de la informática, esta es significativamente grande y en ocasiones confusa;

Pasamos de páginas estáticas hechas con HTML, Java Script, CMS y estándares como XHTML, CSS, P3P por nombrar algunas de ellas.

Todo enfocado hacia el usuario final, a su interacción y producción. La Web crece a medida que los usuarios colaboran usando estas plataformas tecnológicas. Esta evolución tecnológica es lo que muchos llaman ahora la Web 2.0, no es una versión, sino un concepto y una forma diferente de ver las cosas (Escalona, 2002).

2.5. DESARROLLO WEB BASADA EN U. M. L. (U. W. E.)

2.5.1. LENGUAJE UNIFICADO DE MODELADO (U.M.L.)

UML (*Unified Modeling Language*) es un lenguaje gráfico para visualización, especificación, construcción y documentación de componentes de sistemas grandes y complejos, como lenguaje de especificación, significa la construcción de modelos precisos, completos y no ambiguos. En particular, maneja la especificación de todo el análisis, diseño y decisiones de implementación que puede hacerse en la etapa de desarrollo e implantación de un sistema grande y complejo de software y como lenguaje de documentación, maneja la documentación de la arquitectura de un sistema y todos los detalles. Introduce diagramas que representan la parte dinámica de los procesos, logrando de esta manera identificar fallas de diseño en los procesos y por consiguiente generadoras de errores. Permite estereotipar sus elementos para que tengan un comportamiento particular (Pérez, 2010).

Para entender esta metodología, se necesita construir un modelo conceptual del lenguaje y este requiere tres elementos especiales: construcción de bloques en UML, las reglas que indican como esos bloques construidos se pueden interrelacionar, y algunos mecanismos comunes que se aplican en todo el lenguaje para la construcción de bloques en UML se divide en tres categorías que son: cosas, relaciones y los diagramas. UML proporciona un

gran número de diagramas, algunos son más necesarios dependiendo el tipo proyecto (Pérez, 2010).

2.5.2. INGENIERÍA WEB BASADA EN U.M.L. (U.W.E.)

U.W.E. es una metodología que permite especificar de mejor manera una aplicación web, para el proceso de creación de aplicaciones y describe que es lo que se debe utilizar. Este proceso iterativo e incremental, incluye flujos de trabajo y puntos de control, y sus fases coinciden con las propuestas en el proceso unificado de modelado. (Santamarina, 2010)

La metodología U.W.E. define vistas especiales representadas gráficamente por diagrama en UML. Los principales aspectos en los que se fundamente U.W.E. son los siguientes:

- Notación estándar: el uso de la metodología U.M.L. para todos los modelos.
- Métodos definidos: pasos definidos para la construcción de cada modelo.
- Especificación de restricciones: recomendables de manera escrita, para que la exactitud en cada modelo aumente.

2.5.3. FASES DE DESAR ROLLO

Las actividades base de modelado de U.W.E. son el análisis de requerimientos, el modelo conceptual, el modelo navegacional y el modelo de presentación. A estos modelos se pueden sumar otros modelos como son el modelo de usuario, modelo de adaptación y modelo de tareas para representar los aspectos dinámicos de la aplicación mediante la

descripción de situaciones. De esta manera se obtiene una colección de modelos y diagramas que describen una aplicación web de manera integral (Pérez, 2010).

- Análisis de requisitos: plasma los requerimientos funcionales de la aplicación web, mediante modelos de casos de uso.

- Diseño conceptual: se define mediante un modelo de dominio, considerando los requisitos plasmados en los casos de uso, el diagrama de clases representará los conceptos con un gran porcentaje de detalle.

- Diseño navegacional: Comprende la construcción del modelo de navegación en dos pasos:

    - Modelo de espacio de navegación: su objetivo es especificar qué objetos pueden ser visitados a través de la aplicación.
    - Modelo de estructura de navegación: amplía el modelo con un modelo de estructuras de acceso necesarias para la navegación como los índices, consultas y visitas guiadas.

- Diseño de presentación: permite la especificación lógica de la aplicación web. Basada sobre este modelo lógico, una presentación física puede ser construida. Representa las interfaces de usuario por medio de vistas estándares de interacción U.M.L. Dentro de este modelo se distinguen dos diferentes vistas:

    - Estructura de vista: muestra la estructura del espacio de presentación.

- Interfaz de usuario (UI por sus siglas en inglés de *User Interface*): Se refleja los detalles acerca de los elementos de la interfaz de usuario dentro de las páginas.

2.5.3.1 ANÁLISIS DE REQUISITOS

El análisis de requisitos se expresa a través de la especificación de los casos de uso del sistema. Un caso de uso en UML es una unidad coherente de la funcionalidad proporcionada para la aplicación que obra recíprocamente con unos o más actores de la aplicación. Describe una parte del comportamiento de la aplicación sin revelar la estructura interna. (Pressman, 2012)

De esta manera, los requisitos para una aplicación web se pueden especificar con un modelo de casos de uso. Un ejemplo de Casos de Uso se puede ver en la Figura 2.5

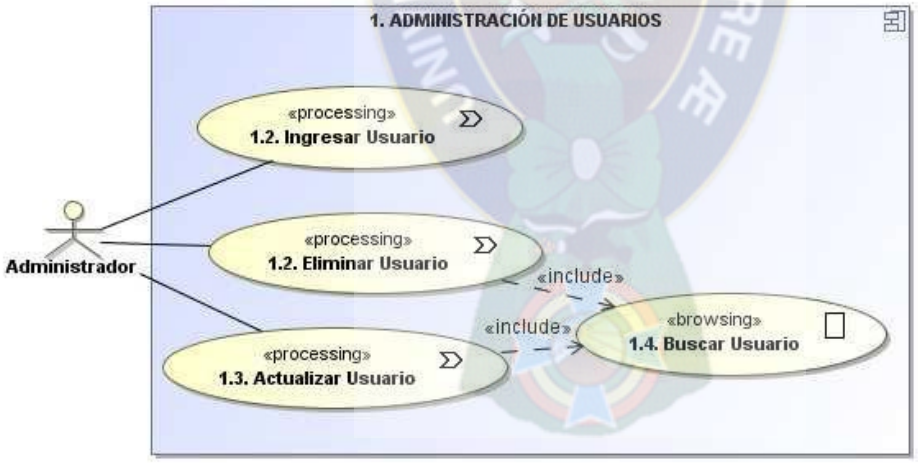


Figura 2. 5. Ejemplo de un Diagrama de Casos de Uso.
Fuente: (Pressman, 2012)

2.5.3.2 DISEÑO CONCEPTUAL

El modelado conceptual para aplicaciones web dentro de UWE no difiere del modelado conceptual de las aplicaciones normales. Se utiliza un diagrama de clases para representar gráficamente un modelo conceptual como visión estática que demuestre una colección de los elementos estáticos del dominio. (Pressman, 2012)

La construcción del modelo conceptual se lleva a cabo de acuerdo con los casos de uso que se definieron en la especificación de los requerimientos de usuario. En la Figura2.6. podemos ver un ejemplo de un modelo conceptual con diagrama de clases.

Figura 2. 6. Ejemplo de un modelo Conceptual – Diagrama de Clases
Fuente: (Pressman, 2012)

El modelo de navegación de una aplicación web comprende la especificación de qué objetos pueden ser visitados mediante la navegación, a través del sistema y las asociaciones entre ellos. Mediante estos diagramas se representa el diseño y la estructura de las rutas de navegación al usuario para evitar la desorientación en el proceso de navegación. Los elementos básicos en el modelo de navegación son los nodos y enlaces como se puede apreciar en la Figura 2.7 (Pressman, 2012)

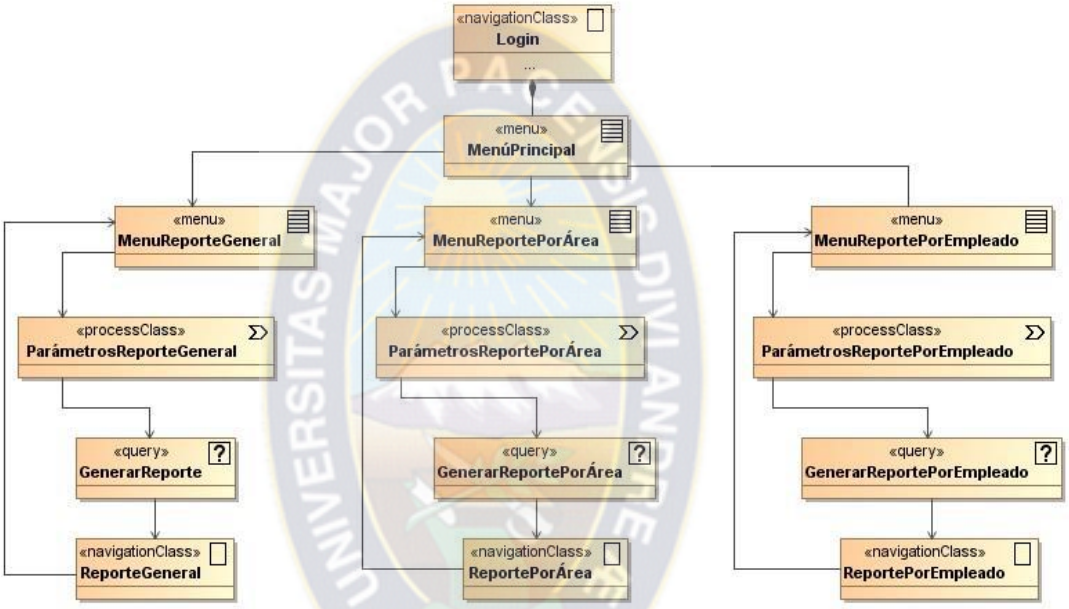


Figura 2. 7. Ejemplo de un modelo de Navegación.

Fuente (Pressman, 2012)

2.5.3.4 DISEÑO DE PRESENTACIÓN

El modelo de presentación en U.W.E. está muy relacionado con los elementos de las interfaces definidas en HTML. Estos elementos también están definidos como estereotipos de UML. Los elementos del modelo de presentación son: ventanas, entradas de texto, imágenes, audio y botones (Pressman, 2012)

Como se puede ver en la a continuación, estos diagramas permiten especificar dónde y cómo los objetos de navegación serán presentados al usuario, una representación esquemática visible para el usuario.

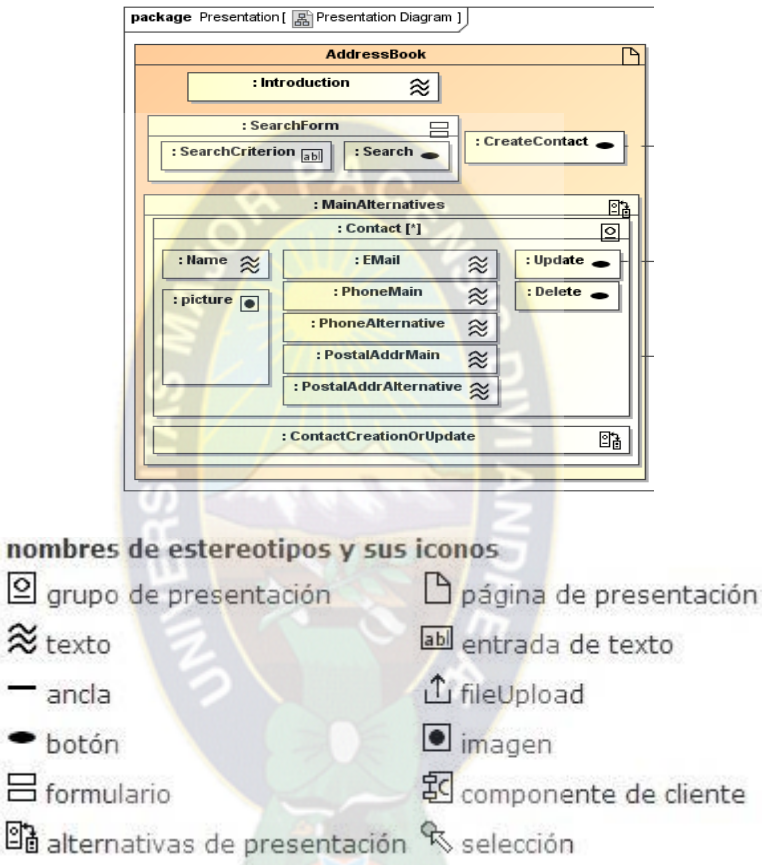


Figura 2. 8. Ejemplo de un Modelo de Presentación.
Fuente: (Pressman, 2012)

2.6. PROCESO DE AUTOEVALUACIÓN

La Autoevaluación corresponde a un proceso de investigación del quehacer académico institucional, cuyo fin es promover su mejoramiento; como resultado de una práctica de autoconocimiento que conlleva cambios o transformaciones coherentes con los principios, propósitos y funciones de la Universidad.

La responsabilidad del proceso, tanto en su organización como ejecución, recae en la entidad académica interesada en autoevaluarse, la cual debe contar con la asesoría del Centro de Evaluación Académica.

Para el desarrollo del presente proyecto se basa en el "reglamento de autoevaluación, evaluación interna y acreditación de la universidad mayor de san Andrés", capítulo 4 artículo 17, inciso a. (Reglamento de autoevaluacion evaluacion interna y acreditacion de la Universidad Mayor de San Andres, 2009)

CAPÍTULO III

MARCO APLICATIVO

3.1. INTRODUCCIÓN

En este capítulo se efectúa el diseño y desarrollo del sistema correspondiente, dado que Scrum es una metodología simple y concisa pero que exige un gran esfuerzo y trabajo duro si se quiere lograr un producto de calidad, tenemos que entender que no se sigue un plan, sino que es un adaptación constante a las necesidades y circunstancias de la evolución y crecimiento del proyecto.

Para poder aplicar la ingeniería de requisitos, se aplica las actividades correspondientes a la conceptualización, análisis, diseño y desarrollo del sistema de acuerdo a un proceso.

Cuando se comienza una iteración ("Sprint") se determina que parte o módulos se van a desarrollar, tomando en cuenta criterios de prioridad para el negocio, y la cantidad total de trabajo que se pondrá durante la iteración.

Para el desarrollo del sistema se escoge la metodología Ágil SCRUM, que utiliza un modelo de desarrollo incremental, y este se complementa con la metodología UML para las etapas de desarrollo en cada iteración. Para tal efecto el proyecto se dividirá en tres módulos:

3.1.1. IDENTIFICACIÓN DE ROLES SCRUM

En la tabla 3.1 se tiene el equipo se focaliza en construir software de calidad implicadas e identificadas en los roles de Scrum para el desarrollo del proyecto (ver Cap. 2.2.1).

Tabla 3. 1 Identificación de Roles Scrum

ROL		NOMBRE
Propietario del Producto		UGICA
Scrum Master		Raquel Yujra
Equipo Scrum	Analista Diseñador Desarrollador Tester	Raquel Yujra

3.2. PRE – GAME

3.2.1. RECOPILACIÓN DE REQUERIMIENTOS

Se estiman las prioridades de los módulos a desarrollar en relación a los requerimientos de los usuarios.

Un *Backlog Eficiente* es aquel donde se puede obtener el mayor beneficio con el menor esfuerzo posible.

Este concepto llevado al *ProductBacklog* significa invertir el esfuerzo de exploración y especificación de la manera más inteligente posible para evitar re-trabajos y desperdicios innecesarios.

Por esto se recomienda un *ProductBacklog* donde sus ítems más prioritarios están expresados con un nivel de detalle mucho mayor que los ítems de menor prioridad, los cuales estarán descriptos a un nivel más alto, ya que son los más susceptibles de ser alterados o reemplazados.

Por todo aquello, a continuación, se menciona algunas características del flujo de trabajo en relación al proceso de autoevaluación.

Mediante una Resolución del Honorable Consejo Facultativo, se ratifica la designación de los miembros de la Comisión de Autoevaluación de la unidad académica a autoevaluar, e inicia el proceso de autoevaluación.

La comisión de autoevaluación se encargará de seleccionar y asigna los componentes y criterios según la información que aportan los agentes.

La comisión auto evaluadora realiza la asignación de pesos porcentuales según actores para poder obtener datos y realizar las estadísticas correspondientes al proceso de autoevaluación.

Por lo expuesto anteriormente se llega a la Tabla 3.2 donde son centralizados los requerimientos establecidos y solicitados por los usuarios mediante reuniones en cada sprint.

Tabla 3. 2. Requerimientos Institucionales

ID	DESCRIPCIÓN	MÓDULO	PRIORIDAD	ESTADO
R1	El usuario puede restaurar su contraseña por correo electrónico.	Autoevaluación	Media	Terminado
R2	El administrador crea, edita, y elimina un componente	Componente y Criterio	Alta	Terminado
R3	El administrador crea, edita, y elimina un criterio	Componente y Criterio	Alta	Terminado
R4	El administrador podrá asignar un valor a un criterio.	Componente y Criterio	Alta	Terminado
R5	El usuario cuenta con control y acceso seguro.	Autoevaluación	Alta	Terminado
R6	El administrador podrá asignar una escala al criterio.	Componente y Criterio	Alta	Terminado
R7	El administrador crea formulario	Formulario y Resultados	Alta	Terminado
R8	El administrador podrá publicar un formulario	Formulario y Resultados	Media	Terminado
R9	El administrador genera reporte de acuerdo con los resultados de la encuesta	Formulario y Resultados	Media	Terminado

3.2.2. MODELADO DE CASOS DE USO

Realizando la refactorización entre las metodologías SCRUM y UWE, se genera:

En la Figura 3.1 se puede ver el diagrama de Caso de Uso de Alto Nivel del sistema a desarrollar:

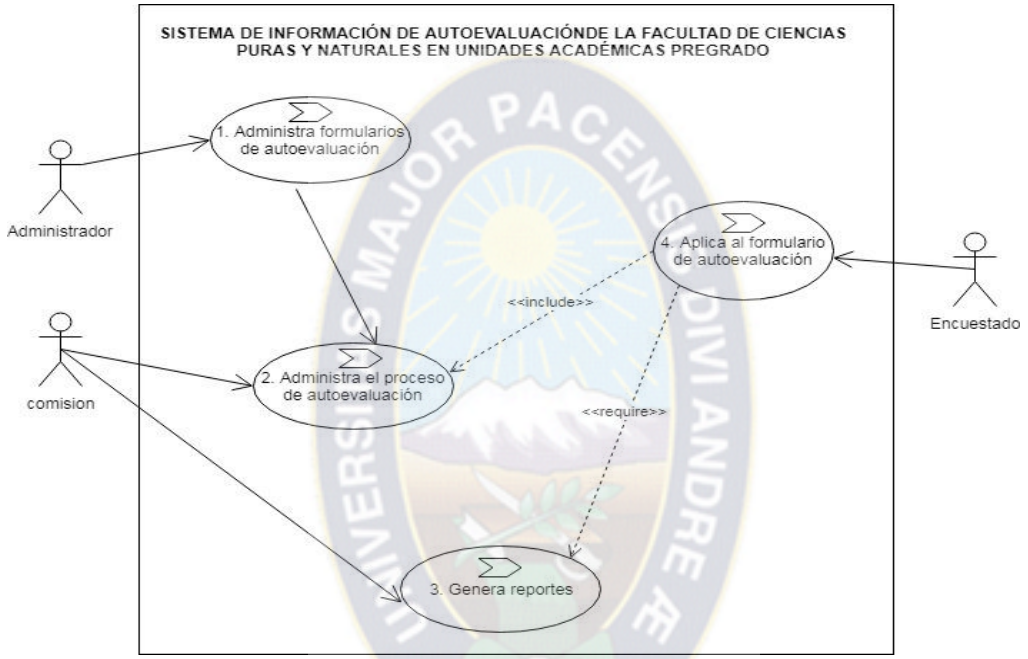


Figura 3. 1. Diagrama de Casos de Uso de Alto nivel del Sistema

Se muestra en el caso de uso para la Administración de los formularios de autoevaluación, la representación de las funciones del administrador en el sistema en la Figura 3.2.

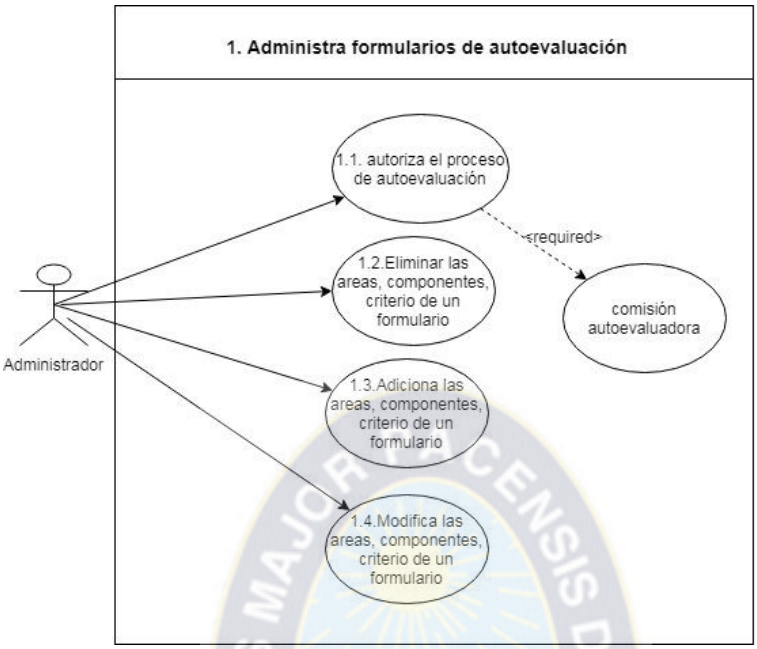


Figura 3. 2. Caso de Uso Administrar formularios de autoevaluación

A continuación, en la figura 3.3 se muestra el diagrama de caso de uso para la administración del proceso de autoevaluación junto con sus especificaciones, y la representación de la administración del inicio del proceso de autoevaluación.

Figura 3. 3. Caso de Uso Administrar los procesos de Autoevaluación.

Se observa a continuación el diagrama de caso de uso que representa la aplicación a los formularios, que son publicados por el administrador, en este caso la comisión autoevaluadora ver la figura 3,5.

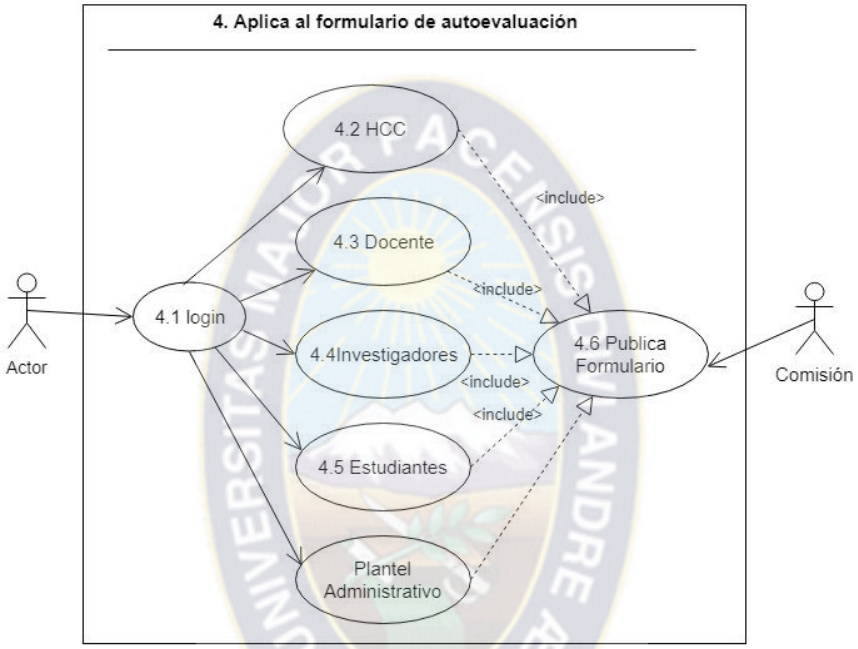


Figura 3. 4. Caso de Uso. Aplica al formulario de autoevaluación

A continuación en la figura 3.5. Se puede observar el caso de uso que representa la publicación de la autoevaluación a los actores correspondientes, la finalización de su aplicación y el poder generar reportes con los resultados obtenidos en la autoevaluación.

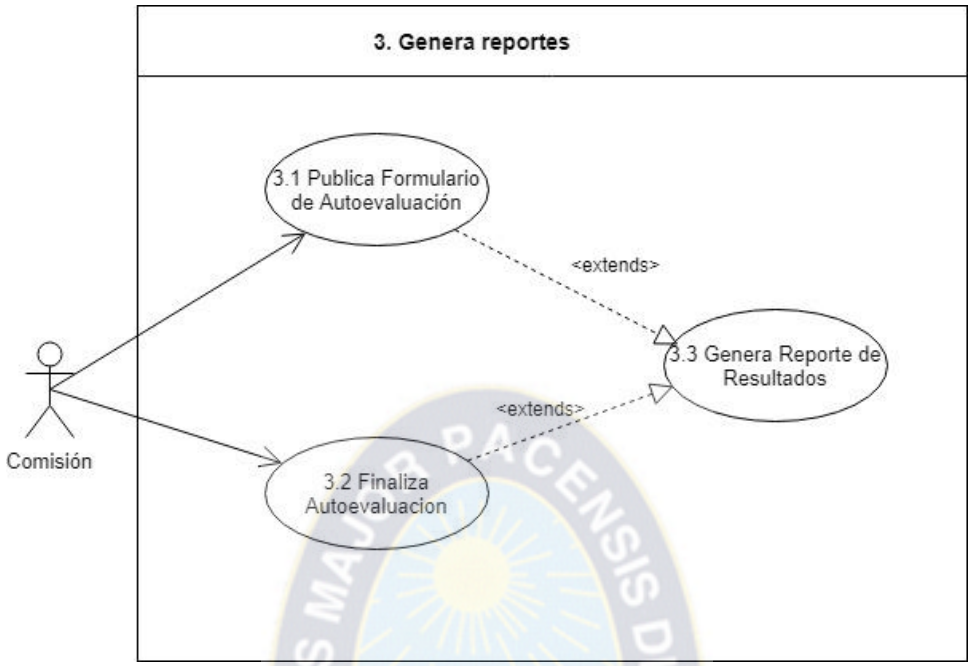


Figura 3. 5. Caso de Uso. Genera reportes

3.2.3. MODELADO DE DIAGRAMA DE SECUENCIA

En la figura 3.6. se muestra el modelo de secuencia de eventos al iniciar el administrador al sistema, que administra a los usurios que igresan al sistena.

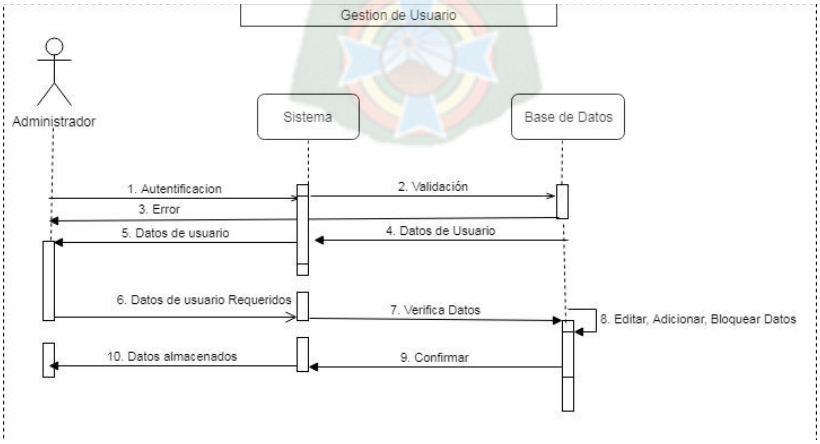


Figura 3. 6. Modelo secuencial del Administrador

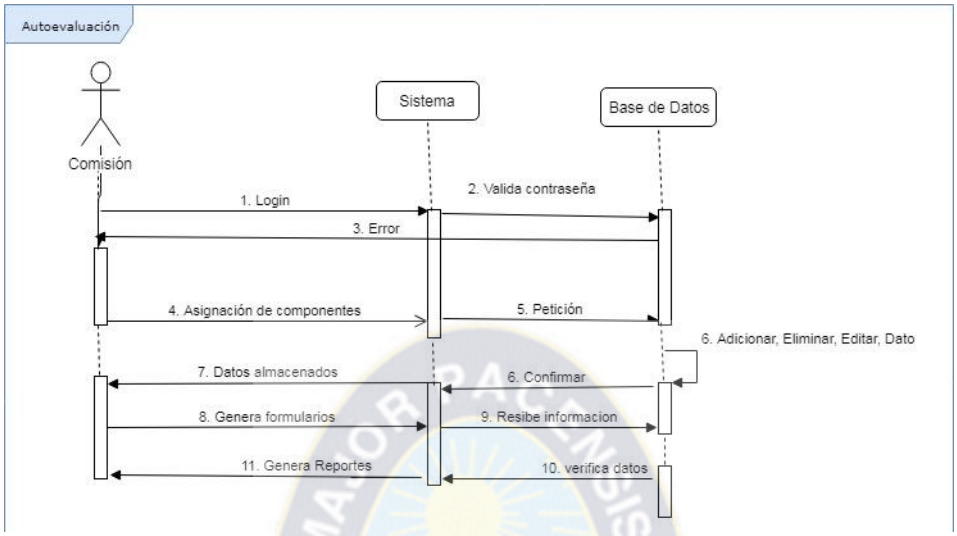


Figura 3. 7. Modelo secuencial del usuario Comisión

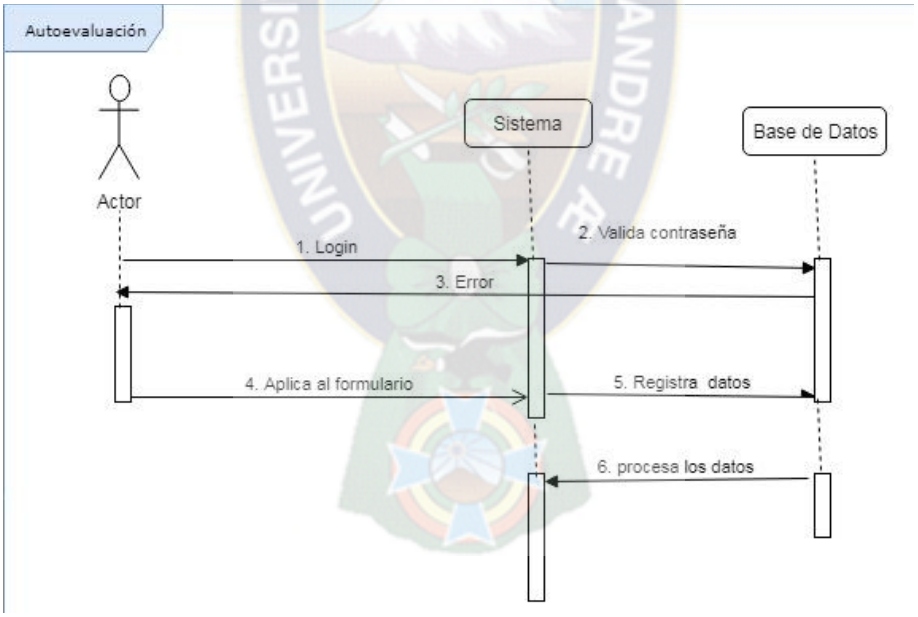


Figura 3. 8. Modelo secuencial del usuario Actor

3.2.4. MODELADO DE DIAGRAMA DE ESTADO

Los estados hacen referencia a una condición durante la vida una interacción durante la cual se satisface alguna condición en la figura se muestra los estados del sistema.

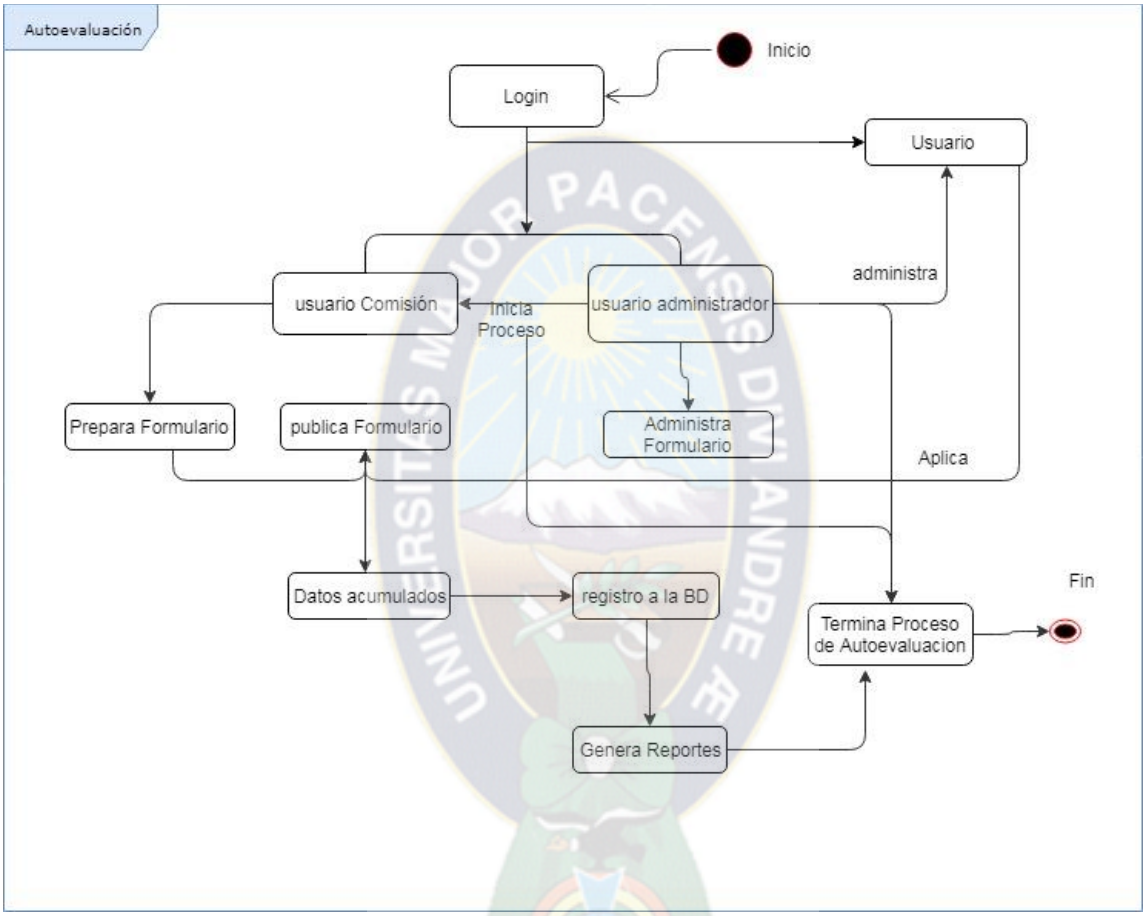


Figura 3. 9. Modelo de diagrama de secuencia del sistema

3.2.5. DEFINICIÓN DEL CRONOGRAMA DE TRABAJO

El cronograma de trabajo se definió en base al ciclo de vida de la metodología SCRUM, y se identifican 3 etapas principales, ya mencionadas anteriormente *pre-game, game y post-*

game, este cronograma se puede apreciar en un diagrama de *Gantt* en los Anexos de este documento.

3.2.6. ANÁLISIS DE RIESGOS

Es la probabilidad de ocurrencia de un acontecimiento adverso, problema o daño y las consecuencias del mismo y el análisis de riesgo es la aplicación de los métodos y herramientas existentes para determinar tal probabilidad, entre ellas tenemos 3 tipos de riesgo:

- Riesgo del Proyecto. Que afecta a la ejecución de los módulos en ciertos periodos de tiempo ya definidos por el grupo de trabajo.
- Riesgos del Producto. Afectan a la calidad o rendimiento del software que se está desarrollando.
- Riesgo de Negocio. Afectan a la Organización o Institución que desarrolla o suministra el Software.

Tabla 3. 3. Análisis de Riesgos

Riesgo	Tipo	Descripción	Probabilidad	Efecto	Estrategia
No se cumplen con las fechas establecidas en el cronograma.	Proyecto	Es probable que las fechas descritas en el diagrama de Gantt no se cumplan.	Alta	Tolerable	Realizar un segundo cronograma más flexible.
Cambios frecuentes en los requerimientos del cliente	Proyecto Producto	El riego más común es generalmente el miedo al cambio en relación a los	Alta	Serio	Realizar reuniones de capacitación constante.

		usuarios finales			
No se cumplen los plazos de entrega del software.	Producto	Los plazos de entrega están sujetos al Scrum Master del proyecto	Moderada	Serio	Agilizar procesos de desarrollo
No existe el incentivo adecuado para la elaboración del proyecto.	Producto Proyecto	Disminución de activos, presión innecesaria.	Baja	Tolerable	Entregas de versiones funcionales en cortos periodos de tiempo.

3.3. GAME

Durante esta etapa del proyecto cada iteración adoptó los modelos de la metodología UWE que posteriormente fueron implementados basándose principalmente en la Base de Datos de Software (Gestor de Base de Datos MySQL).

En cuanto a la persistencia de objetos se optó por hacer corresponder cada clase del modelo conceptual con alguna tabla de la Base de Datos, donde las filas representarían las instancias de los objetos y las columnas a los atributos de las clases.

Las clases y sus métodos fueron implementados en el lenguaje *PHP*

Finalmente se desarrollaron las vistas de entornos Web en base a los modelos de presentación.

3.3.1. PRIMERA ITERACIÓN

Durante la primera iteración se desarrollan los elementos necesarios y la base para los módulos del proceso de autoevaluación y su administración. Las actividades realizadas durante esta iteración se observan en la siguiente tabla, que constituye el *BackLog del Sprint*.

Tabla 3. 4. Backlog del primer sprint

		Sprint	Inicio	Duración
		1	1/02/2018	28/02/2018
ID	**Tareas**	**Tipo**	**Días de trabajo**	**Estado**
1.1	Realizar la planificación de la iteración.	Planificación	2	Terminado
1.2	Analizar los requerimientos del Backlog del producto.	Planificación	2	Terminado
1.3	Analizar los requerimientos de la interacción con historias de usuario	Desarrollo	2	Terminado
1.4	Diseñar la base de datos	Desarrollo	5	Terminado
1.5	Construir el módulo central de administrativos.	Desarrollo	3	Terminado
1.6	Construir el módulo central de estudiantes.	Desarrollo	3	Terminado
1.7	Construir el módulo central Asignación de Componentes	Desarrollo	3	Terminado
1.8	Construir el módulo central de acceso al sistema	Desarrollo	3	Terminado
1.9	Construir el módulo central	Desarrollo	3	Terminado
1.10	Probar los módulos desarrollados en la iteración.	Desarrollo	1	Terminado

Funcionalidades correspondientes al incremento de la iteración son:

- Base de Datos independientes del sistema
- Etiquetas realizadas para el entorno grafico de sistema de registro y administración de Archivos

- Página de ingreso con control de acceso a los usuarios
- Módulo de registro usuarios nuevos
- Módulo de administración del sistema.

3.3.1.1 MODELO CONCEPTUAL

Se genera el modelo conceptual en base a la estructura final de la base de datos considerando el manejo de consultas en un modelo relacional.

A continuación se presenta el modelado de diagramas de clases donde se muestran los modelos que se utilizaron y ejecutaron en el sistema, este diagramas de clases es el pilar fundamental para el módulo de asignación de componentes según la información que aportan los agentes evaluadores ver la Figura 3.10.

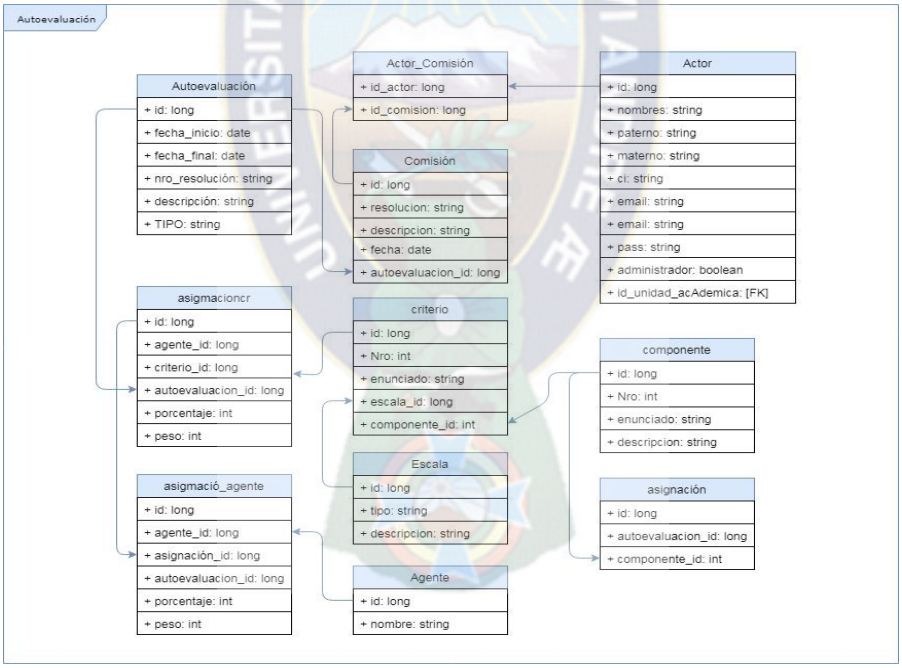


Figura 3. 10. Modelo conceptual del módulo asignación de componentes según los actores

A continuación observamos en la Figura 3.11 el módulo de la creación del formulario para la aplicación de los actores.

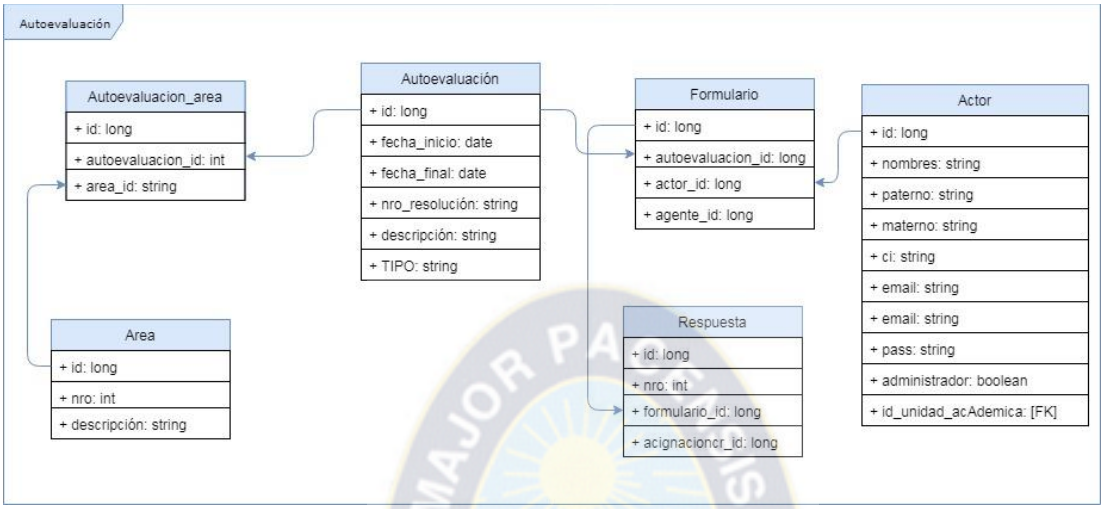


Figura 3. 12. Modelo conceptual Módulo de la generación de formularios

3.3.1.2 MODELO DE NAVEGACIÓN

La Figura 3.13. Muestra el modelo de navegación en base a la pila del Sprint 1.

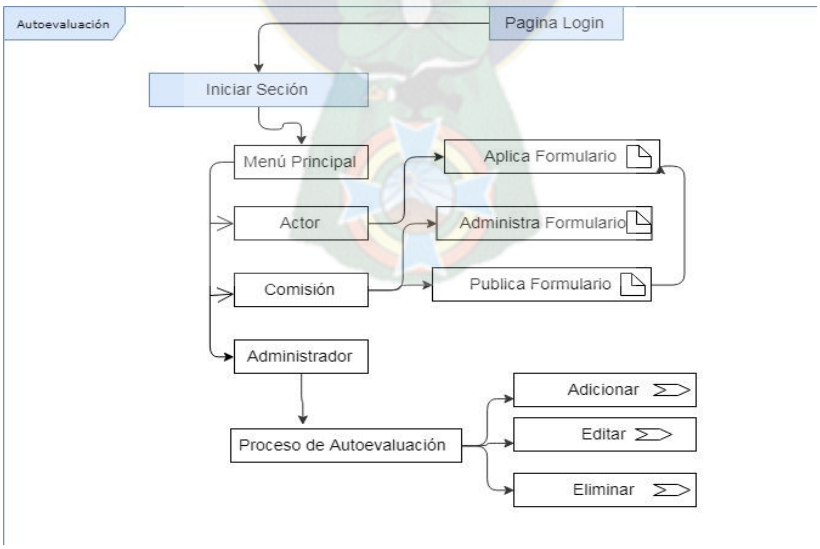


Figura 3. 13. Modelo navegacional para el Sprint 1

3.3.1.3 MODELO DE PRESENTACIÓN

En la Figura 3.14 se muestra el modelo de presentación para el módulo central de administrativos.

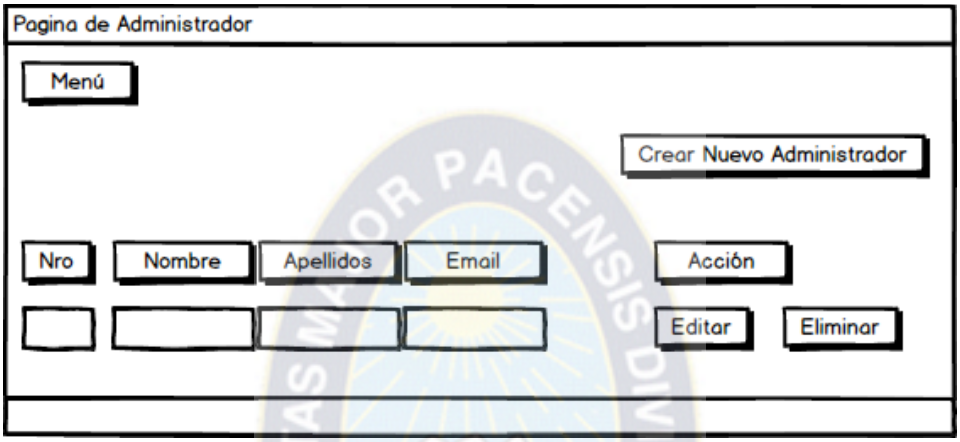


Figura 3. 14. Modelo de presentación módulo central de administrativos

A continuación en la Figura. 3.15 se puede ver la representación del módulo de presentación para administrar los procesos de autoevaluación.

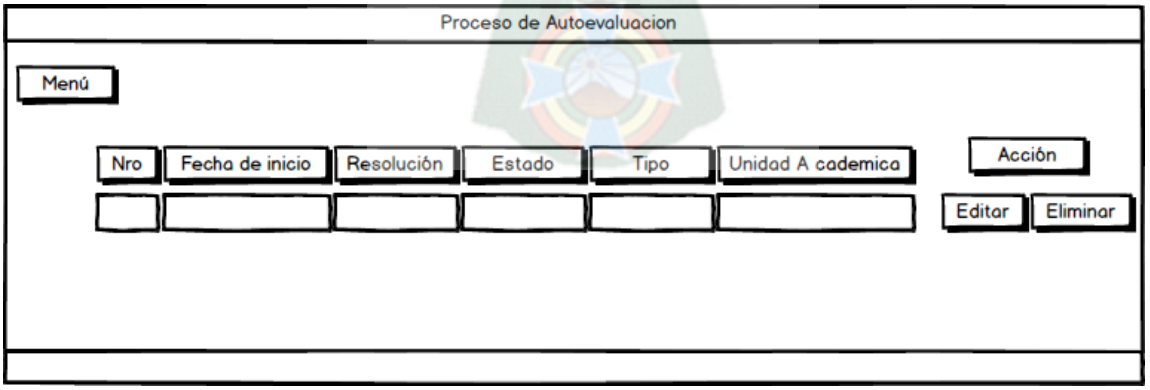


Figura 3. 15. Modelo de presentación módulo de proceso de autoevaluación

Se observa la representación del módulo de administración de componentes y criterios en la Figura. 3.16

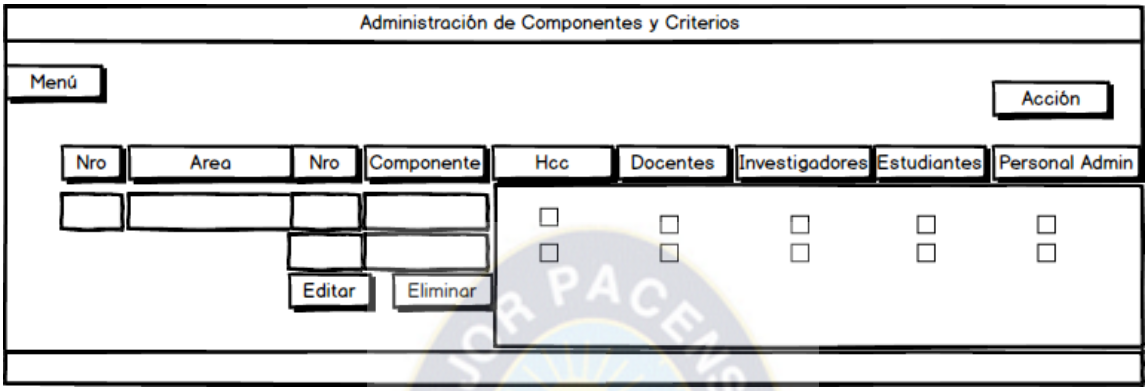


Figura 3. 16. Modelo de presentación módulo de administración de componentes y criterios

Por último se observa la representación del módulo de resultados y los formularios administrados y preparados por la comisión, ya publicados en la Figura 3.17.

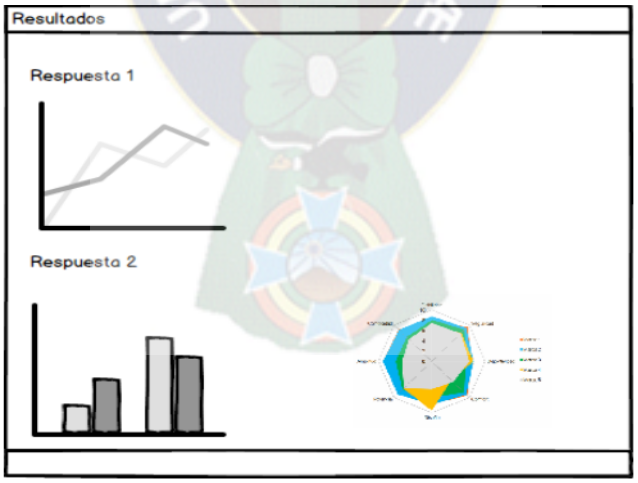


Figura 3. 17. Modelo de presentación módulo Formularios y Resultados

3.3.2. SEGUNDA ITERACIÓN

En esta iteración se desarrolló el módulo del curso de temporada tanto para el administrador como para los usuarios.

		Sprint	Inicio	Duración
		2	1/04/2018	21/05/2018
ID	**Tareas**	**Tipo**	**Días de trabajo**	**Estado**
1.1	Realizar la planificación de la iteración.	Planificación	2	Terminado
1.2	Analizar los requerimientos del Backlog del producto.	Planificación	2	Terminado
1.3	Analizar los requerimientos de la interacción con historias de usuario	Desarrollo	2	Terminado
1.4	Construir el módulo de administración de los formularios	Desarrollo	3	Terminado
1.5	Construir el módulo de generación de reportes.	Desarrollo	3	Terminado
1.6	Probar los módulos desarrollados en la iteración.	Desarrollo	1	Terminado

Tabla 3. 5: Backlog del segundo sprint

3.3.2.1 MODELO DE NAVEGACIÓN

La figura 3.18 muestra el modelo navegacional en base a la pila del sprint 2.

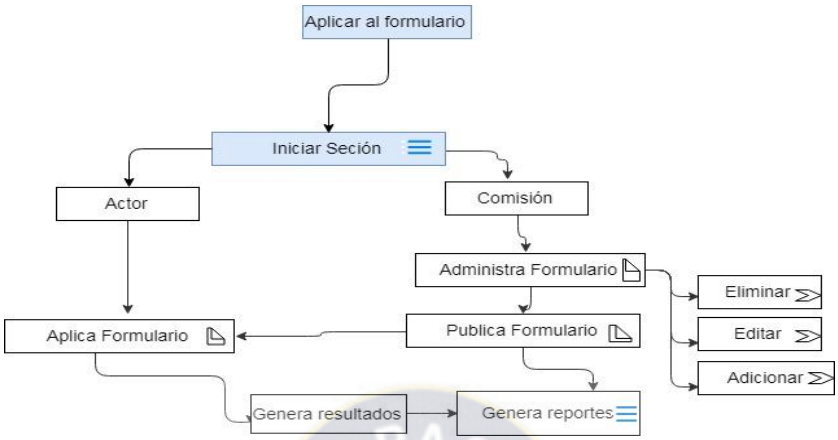


Figura 3. 18: Modelo navegacional para el Sprint 2

3.3.2.2 MODELO DE PRESENTACIÓN

En la Figura 3.19 se ve el modelo de presentación para el módulo administración del curso de temporada.

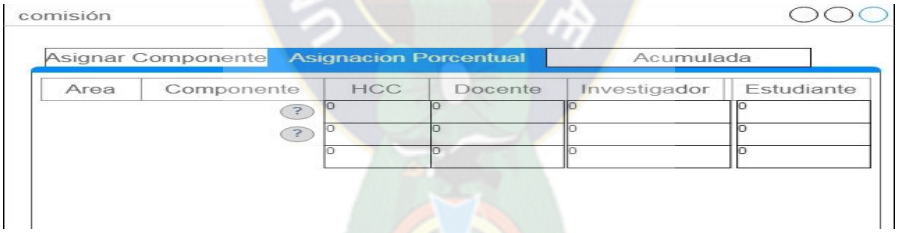


Figura 3. 19: Modelo de presentación administración de la asignación de porcentajes

En la Figura 3.20 se ve el modelo de presentación para el módulo del cálculo general según percepción de quienes evalúan.

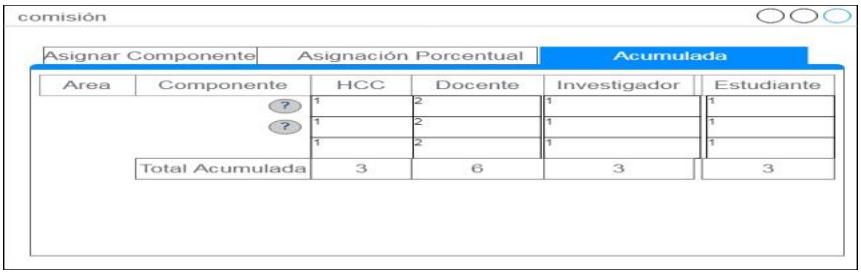


Figura 3. 20: Modelo de presentación módulo calculo general según percepción de quienes evalúan

En la Figura 3.21 se ve el modelo de presentación para el módulo escalas multidimensionales de los resultados obtenidos en la encuesta.

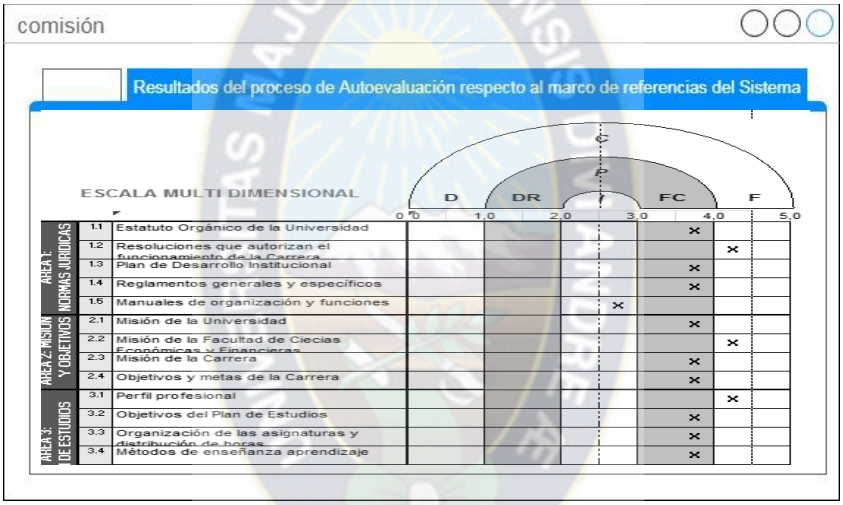


Figura 3. 21: Modelo de presentación módulo escalas multidimensionales

A continuación, se tiene las capturas de pantalla de las tareas implementadas en el Sprint 2.

En la Figura 3.21 se muestra la pantalla de registro de componentes.

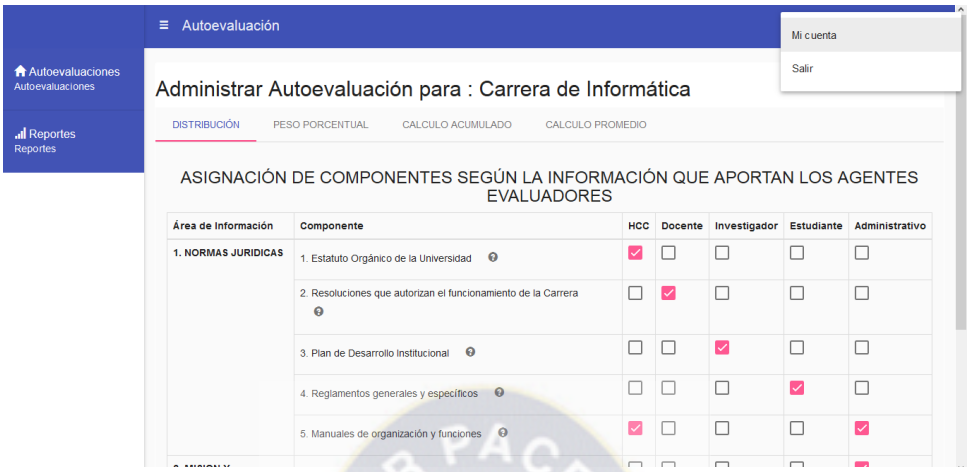


Figura 3. 22: Modelo de presentación de asignación de componentes

En la Figura 3.23 se muestra la pantalla del cálculo acumulado según el actor.

Figura 3. 23: Pantalla del cálculo acumulado según el actor

En la Figura 3.24 se muestra la pantalla de resultados del proceso de Autoevaluación respecto al marco de referencias del Sistema de la Universidad Boliviana

Figura 3. 24: Pantalla resultados del proceso de Autoevaluación respecto al marco de referencias del Sistema de la Universidad Boliviana

En la Figura 3.25 se muestra la pantalla de resultados y gráficas.

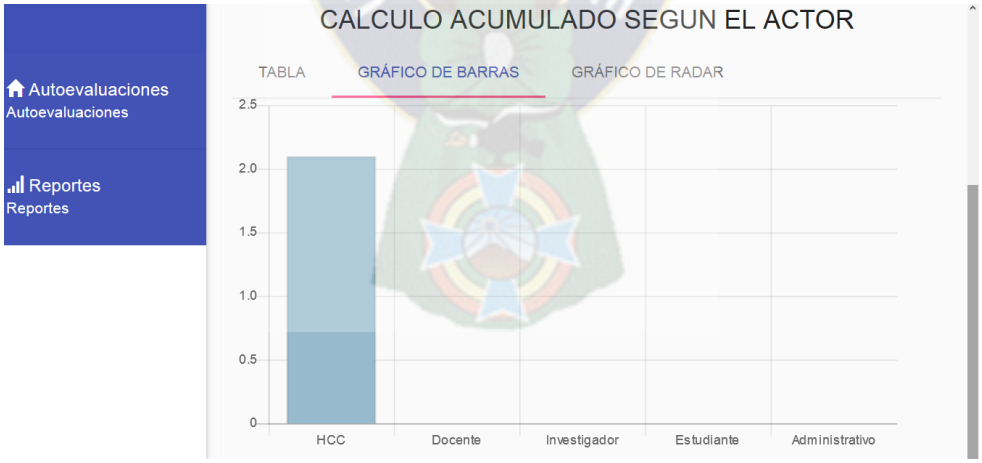


Figura 3. 25: Pantalla de resultados y graficas

En la Figura 3.26 se muestra la pantalla de la aplicación al formulario según los actores, las áreas y componentes habilitados.

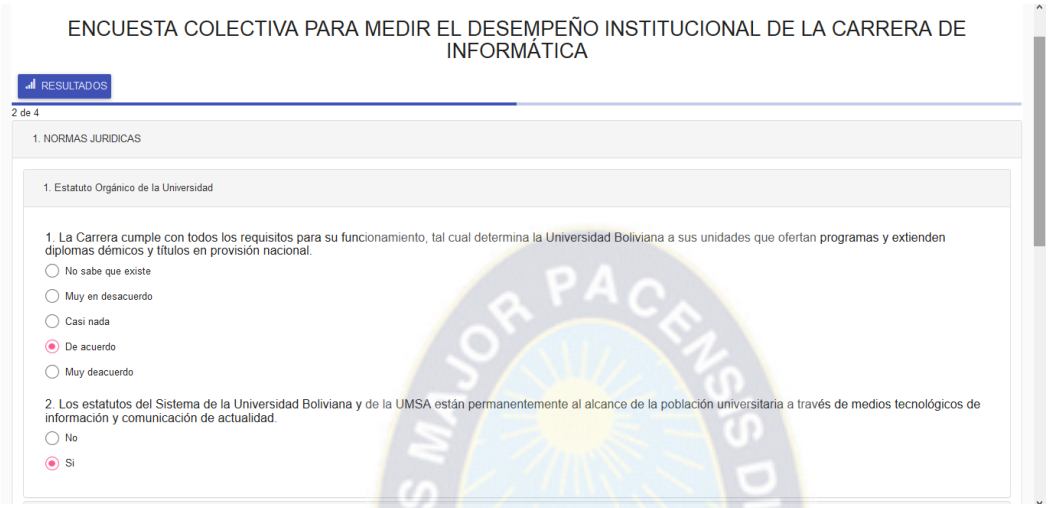


Figura 3. 26: Pantalla de formularios

3.4. POST – GAME

En esta etapa final se realizaron las actividades de prueba del Software Web, se propusieron métricas de calidad que se pueden apreciar en el Capítulo IV y se desarrollaron los diferentes manuales de usuario.

La etapa de pruebas de un sistema presenta un elemento crítico para la garantía de la calidad de software, y representa una revisión final de las especificaciones del diseño y la codificación (Pressman, 2012).

Para hacer las pruebas del sistema se hizo uso de las pruebas de prototipado rápido, donde se centra en una representación de aquellos aspectos del software que serán visibles para el

cliente o el usuario final. Este diseño conduce a la construcción del prototipo, el cual fue evaluado por la institución para una retroalimentación, gracias a esto se refinan los requisitos del software que se desarrolló.

La interacción ocurrirá hasta que el sistema se ajuste para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo que el desarrollador entienda lo que debe hacer y el cliente vea resultados en corto plazo.

- Perfectivo: son las acciones llevadas a cabo para mejorar la calidad interna de los sistemas en cualquiera de sus aspectos: reestructuración del código, definición más clara del sistema y optimización del rendimiento y eficiencia.

- Evolutivo: son las incorporaciones, modificaciones y eliminaciones necesarias en un producto para cubrir la expansión o cambio en las necesidades del personal.

- Adaptativo: son las modificaciones que afectan a los entornos en los que el sistema opera, por ejemplo, cambios de configuración del hardware, software de base, bases de datos, comunicaciones, entre otros.

- Correctivo: son aquellos cambios precisos para corregir errores del producto software.

También consideramos las pruebas de stress a los servicios HTTP que el *back-end* proporciona, estas pruebas fueron realizadas con *Jmeter* que se dedica a realizar todo tipo de pruebas a sistemas.

Se consideró realizar la prueba al servicio de inicio de sesión por ser el más utilizado, teniendo como datos de prueba:

- 100 usuarios conectados al mismo tiempo.
- 30 peticiones por usuario.

La configuración para la prueba se puede ver en la figura 3.27

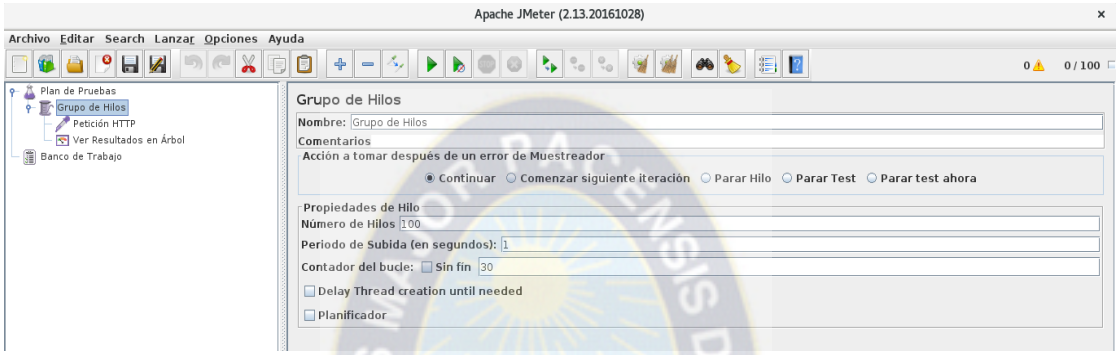


Figura 3. 27. Configuración para pruebas con Jmeter

Las pruebas fueron realizadas directamente en el servidor oficial del proyecto, como se ve en la figura 3.28.

Con la configuración previa se llegó a obtener los resultados mostrados

Figura 3. 28. Configuración para pruebas al servidor oficial

Los datos generados fueron: de las 100 peticiones ninguna se perdió y su media en milisegundos es de 538 de respuesta por petición, mostrando que las respuestas son casi inmediatas de los servicios.

CAPÍTULO IV

CALIDAD Y SEGURIDAD DEL SOFTWARE

4.1. CALIDAD

El objetivo es llegar a la mejor calidad necesaria de evaluación bajo la norma ISO 9126, para ello se realiza los cálculos referentes a la funcionalidad, usabilidad, mantenibilidad y portabilidad del sistema.

4.1.1. FUNCIONALIDAD

La funcionalidad del Sistema se mide a través de su complejidad, la funcionalidad no puede ser medida directamente, entonces corresponde derivar, mediante otras medidas directas como Punto Función, que permite un resultado medible y cuantificable a partir de la siguiente fórmula: Para la medición de la funcionalidad, se toman en cuenta las siguientes características:

Tabla 4. 1: Cálculo Punto Función

Parámetros de medida	Cuenta	Factor de peso			Total
		Simple	Medio	Complejo	
Número de entradas de Usuario	26	3	**4**	6	104
Número de Salidas de Usuario	13	4	**5**	7	65
Número de Peticiones de Usuario	8	3	**4**	6	32
Número de Archivos	23	7	**10**	15	230
Número de Interfaces	0	5	**7**	10	0
				Total	431

Los valores de la variable Fi, se obtiene de los resultados que se aprecian en la Tabla 4.3, bajo la siguiente ponderación:

Tabla 4. 3: Valores de ajuste de complejidad

Escala	Complejidad
0	Sin influencia
1	Incidental
2	Moderado
3	Medio
4	Significativo
5	Esencial

Preguntas para la variable Fi se muestra en la tabla 4.4.

Tabla 4. 4: Ajuste de Complejidad Punto Función

Pregunta	Ponderación
¿Requiere el Software copias de seguridad y de recuperación fiables?	5
¿Se requiere comunicación de datos?	5
¿Existen funciones de procesamiento distribuido?	3
¿Es crítico el rendimiento?	3
¿Se ejecuta el Software en un entorno operativo existente y fuertemente utilizado?	5
¿Requiere el Software entrada de datos interactiva?	3
¿Requiere la entrada de datos interactiva para que estas transacciones de entrada se lleven a cabo sobre múltiples pantallas u operaciones?	5
¿Se actualizan los archivos maestros de forma interactiva?	5

¿Son complejos las entradas, las salidas, los archivos o las peticiones?	3
¿Es complejo el procesamiento interno?	3
¿Se ha diseñado el código para que sea reutilizable?	5
¿Están incluidas en el diseño la conversión y la instalación?	4
¿Se ha diseñado el Software para soportar múltiples instalaciones en diferentes organizaciones?	5
¿Se ha diseñado la Aplicación Web para facilitar los cambios y para ser fácilmente utilizada por el usuario?	5
P(Fi) Total	59

Empleando la fórmula para hallar el PF y PF (Máximo):

$$PF = 431 * [0{,}65 + 0{,}01 * 59]$$
$$PF = 534{,}44$$
$$PF(\text{Máximo}) = 431 * [0{,}65 + 0{,}01 * 70]$$
$$PF(\text{Máximo}) = 581{,}85$$

Con los valores máximos de ajuste de complejidad de Punto Función, se tiene el siguiente resultado de Funcionalidad Real:

$$\text{FUNCIONALIDAD} = \left(\frac{534{,}44}{581{,}85}\right) * 100$$
$$\text{FUNCIONALIDAD} = 91{,}85\ \%$$

Por tanto, la funcionalidad del Software se representa por el 91.85%, teniendo en cuenta el Punto Función (Máximo). Lo que quiere decir que el Software cumple con los requisitos funcionales de forma satisfactoria.

Es decir que en una escala de trabajo de 8 hrs. el software es capaz de ser operable aproximadamente 7 hrs. sin que este se caiga por razones de funcionalidad interna

4.1.2. USABILIDAD

Cuando se habla de usabilidad, se espera que el Software sea de fácil entendimiento y aprendizaje. Cabe resaltar que para la norma ISO 9126, la usabilidad no es afectada por la funcionalidad y eficiencia. La usabilidad está definida por los usuarios finales o asociados al Software.

Para la medición de la usabilidad se tiene el siguiente cuestionario con los ajustes necesarios que se pueden observar en las siguientes tablas:

Tabla 4. 5: Escala de ajuste de usabilidad

Descripción	Escala
Pésimo	1
Malo	2
Regular	3
Bueno	4
Muy bueno	5

Las preguntas utilizadas para medir la funcionalidad fueron las representadas en la tabla 4.6, con los siguientes valores

Tabla 4. 6: Evaluación de usabilidad

Factor	Valor
¿Se ha cubierto todos los requerimientos establecidos por el Software?	5
¿Es sencillo acceder a los datos?	5
¿Presenta suficiente ayuda durante el tiempo que accede al software?	4
¿Los informes son suficientemente representativos?	5

¿El Software tiene la seguridad necesaria?	5
¿Está de acuerdo con el funcionamiento del Software?	4
¿El Software facilita el trabajo que realiza?	5
Total	33

La usabilidad se calcula por medio de la siguiente fórmula:

$$\text{USABILIDAD} = \frac{\left[\left(\frac{\sum Valor}{n}\right) \times 100\right]}{5}$$

$$\text{USABILIDAD} = \frac{\left[\left(\frac{33}{7}\right) \times 100\right]}{5}$$

$$\text{USABILIDAD} = 94.28\ \%$$

Por lo tanto, la usabilidad del software corresponde a un 94.28%, que se interpreta como la facilidad del usuario al interactuar con las interfaces.

Es decir que de 100 usuarios 94 fueron capaces de entender y operar el software, con un nivel de atracción alto en cuanto a la interfaz de usuario.

4.1.3. MANTENIBILIDAD

Es el esfuerzo necesario para realizar modificaciones específicas. Es un conjunto de atributos relacionado con la facilidad de extender, modificar o corregir errores en el sistema. Para calcular la mantenibilidad del sistema se determina mediante la siguiente formula.

$$\text{IMS} = \frac{[Mt - (Fa + Fc + Fd)]}{Mt}$$

Dónde:

- Mt: es el número de módulos de la versión actual.
- Fc: es el número de módulos que se han cambiado.
- Fa: es el número de módulos en la versión actual que se han añadido.
- Fd: es el número de módulos que se han borrado en la versión actual.

Donde los valores del sistema son los siguientes:

$$\text{Mt} = 3, \quad \text{Fc} = 0, \quad \text{Fa} = 1, \quad \text{Fd} = 0$$

Por lo tanto el resultado IMS es:

$$\text{IMS} = \frac{[3 - (0 + 0 + 0)]}{3}$$

$$\text{IMS} = [3 / 3] = \mathbf{1{,}00}$$

Por lo tanto el valor de IMS es de 100%

$$\text{MANTENIBILIDAD} = 100\ \%$$

Al obtener un 100% aproximadamente en mantenibilidad podemos asegurar que el producto de Software contempla una cantidad de esfuerzo mínimo para cualquier cambio o modificación que requiera el Software.

Por lo tanto, existe una probabilidad de un 100 % de que un desarrollador pueda realizar cambios con facilidad en el código o estructura interna del software propuesto.

4.1.4. PORTABILIDAD

El hecho de que el producto Software esté desarrollado en el Lenguaje NodeJs y es en el entorno web nos ayuda de sobremanera en cuanto a su portabilidad, dado que únicamente es necesario tener un navegador instalado, por lo tanto la portabilidad es el de un 100%.

$$\text{PORTABILIDAD} = 100\ \%$$

4.1.5. CALIDAD TOTAL

Para poder obtener la calidad total del Software, sacaremos la media de todas las medidas expresadas en porcentajes.

Tabla 4. 7: Resultados Calidad Total norma ISO 9126

Características	Resultados
Funcionalidad	91,85
Usabilidad	94,28
Mantenibilidad	100
Portabilidad	100
Total	96,56

La calidad total del Software corresponde al 96,56%, lo que se interpreta como la satisfacción que tiene el usuario al interactuar con el Software. En otras palabras, por cada 100 usuarios que manejaron el Software 96 a 97 quedaron satisfechos.

4.2. SEGURIDAD

4.2.1. POLÍTICAS DE SEGURIDAD (USUARIOS)

- Administrador. Puede acceder a todos los módulos del sistema, y tiene permisos de adición, actualización y eliminado de todos los registros.

  Además, es el encargado de imprimir los reportes los resultados obtenidos de la autoevaluación.

- Los actores, que llenan los formularios se registran con su matrícula o ítem y cedula de identidad.

- La comisión encargada de la autoevaluación solo puede asignar componentes a los actores, y puede designar los criterios, con esos datos genera el formulario.

4.2.2. POLÍTICAS DE SEGURIDAD ACCESO AL SOFTWARE

El sistema verifica la autenticidad del usuario y su contraseña, Los usuarios deben de guardar de forma segura su contraseña.

En caso de olvido este puede ser restaurado mediante el correo electrónico asociado al usuario.

El acceso a la base de datos es restringido para usuarios no autorizados mediante contraseña y nombre de usuario en el Servidor.

4.2.3. ENCRIPTACIÓN DE CONTRASEÑAS

Es recomendable que las contraseñas de los usuarios estén encriptados, de esta forma se tiene asegurado que las acciones sean únicamente responsables de los usuarios correctamente identificados. Para el sistema desarrollado se utilizó una función de NodeJs para encriptar cadenas de texto que hace uso de algoritmo de encriptación MD5 (Message digest 5).

Además, para la solicitud de datos por los servicios proporcionados se mantiene la autenticación del usuario mediante un token encriptado principal que es enviado por la cabecera de la petición a través del protocolo HTTPS, así mantenemos la seguridad en las peticiones.

4.2.4. CONTROL DE ACCESOS Y RESTRICCIONES

Se desarrolló una funcionalidad para el control de accesos a ciertas partes del sistema y restricciones de acciones de acuerdo al rol que tenga asignado un usuario.

CAPÍTULO V

5.1. ANÁLISIS DE COSTOS

En este punto cuantificaremos la inversión de los recursos que se empleó en el desarrollo del Software. Para el cálculo de esfuerzo y costo de desarrollo se utilizará el modelo COCOMO. II

ANÁLISIS COSTO – BENEFICIO

5.2. INTRODUCCIÓN

En la realización del presente proyecto, se tomó en consideración tres carreras de la Facultad de Ciencias Puras y Naturales, en primer lugar la carrera de matemática que inicia su proceso de autoevaluación el año 2000 donde la base de datos original se guardó en diskette separada en cuatro tablas en formato de Microsoft Excel la misma que para su procesamiento fue llevada al formato de SPSS. Un listado parcial de los datos podemos observar en las siguientes tablas, en donde, de acuerdo a los formularios de la encuesta, que se realizó en la carrera.

Por otra parte la Carrera de Estadística inició el proceso de autoevaluación y acreditación a fines del año 2010 y concluyó el 2014. Este año se realizó una primera autoevaluación con pares nacionales y en fechas 24, 25 y 26 de noviembre una comisión de pares evaluadores internacionales y nacionales efectuaron la segunda y última evaluación a la carrera. No se utilizó un sistema de apoyo

La Carrera de Informática de la Facultad de Ciencias Puras y Naturales, inicio el proceso de Autoevaluación desde la gestión 2012, permitiendo consolidar la cultura de la Autoevaluación y Autorregulación Institucional, fortaleciendo la credibilidad y el

reconocimiento por parte de la comunidad, en el cual desarrollo un sistema para el apoyo de la autoevaluación de la carrera.

Dado esta información a lo largo de los años no se tuvo un sistema que se desarrolle de forma genérica para cualquier Carrera de la Facultad de Ciencias Puras y Naturales, y el presente proyecto beneficia a la facultad en general.

5.2.1. MÉTRICAS DE SOFTWARE

Se utilizó como punto función PF = 534,44 que se obtuvo en el apartado 4.1.1.

A continuación, realizamos la conversión de punto función a miles de líneas de código mediante la siguiente tabla 5.1:

Tabla 5. 1: Factor LCD/PF de lenguajes de programación

Fuente: [QSM, 2017]

Lenguaje de Programación	**Factor LDC/PF**
Java	53
JavaScript	**47**
Visual Basic	46
ASP	36
Visual C++	34
PHP	12
Ensamblador	320
C	150

Ahora aplicamos la fórmula para calcular las líneas de código.

KLDC: Número estimado de líneas de código distribuidas.

LCD: PF * Factor LDC

LCD = 534, 44 * 47= 25118, 68

Las líneas de código en su totalidad son 25118,68, de las cuales se estima que aproximadamente un 90% del código es reutilizable, en ese entendido el LCD total es:

KLCD = (LCD – LCD reutilizable) /1000

KLCD = (25118,68 – 22606,81) /1000

Por tanto (Miles de líneas de código) KLDC = 2,51

5.2.2. ESTIMACIÓN DEL ESFUERZO

El esfuerzo necesario para concretar un proyecto de desarrollo de software, cualquiera sea el modelo empleado, se expresa en meses/persona (PM) y representa los meses de trabajo de una persona full-time, requeridos para desarrollar el proyecto.

El modelo usado para calcular el esfuerzo es el propuesto por COCOMO.

$$E = a_b(KLDC)^{bb}, en\ peronas\ por\ mes$$

E: Esfuerzo aplicado por persona (KLDC: Número líneas de código)

$$D = C_b(E)^{dd}, en\ meses$$

D: Tiempo de desarrollo en meses

Se utiliza para obtener una primera aproximación rápida del esfuerzo y hace uso de la siguiente tabla de constantes para calcular distintos aspectos de costes:

Se debe de considerar el tipo de proyecto adecuado para el cálculo de las constantes, entre los cuales tenemos:

- Orgánicos. Relativamente sencillos y pequeños, en los que trabajan equipos pequeños con experiencia, sobre un conjunto de datos poco rígidos.
- Semi-acoplados. Proyectos intermedios (en tamaño y complejidad) en los que participan equipos con variados niveles de experiencia, deben satisfacer requisitos medio rígidos.
- Empotrados. Proyectos bastantes complejos, en los que apenas se tienen experiencia y se engloban en un entorno de gran innovación técnica. Además, se trabaja con unos requisitos muy restrictivos y de gran volatilidad ver Figura 5.2.

Tabla 5. 2: Modelo básico para tipos de proyectos.
Fuente: [Gomez y Migani, 2007]

Proyecto de software	a_a	b_b	c_b	d_b
Orgánico	2.4	1.05	2.5	0.38
Semi-acoplado	**3**	**1.12**	**2.5**	**0.35**
Empotrado	3.6	1.2	2.5	0.32

Aplicando las fórmulas anteriormente descritas y tomando en cuenta un tipo de proyecto orgánico se tiene lo siguiente:

$$E = 3 * (2{,}51)^{1.12} = 8{,}4$$

$$D = 2.5 * (8{,}4)^{0.35} = 5{,}27 = 5$$

El personal requerido será igual a: Número de programadores = E/D

Por tanto: Número de Programadores = 8,42 / 5,27 = 1,59 = 2

5.2.3. COSTO DE SOFTWARE

Tomando como base el salario aproximado de un programador 500 USD, esta cifra se toma en cuenta para estimar el costo del software:

Costo Soft = Nro Programadores*salario Programador*duración=2*500 USD*6= 6.000$

5.2.4. COSTO DE ELABORACIÓN DEL PROYECTO

Los costos de elaboración del proyecto se detallan a la siguiente Tabla 5.3 a continuación:

Tabla 5. 3: Coste de elaboración

Descripción	Costo Total (USD)

Análisis y Diseño del proyecto	500
Bibliografía	30
Material de escritorio	40
Otros	30
Total	600

5.2.5. COSTE DE IMPLEMENTACIÓN DEL PROYECTO

La Facultad como parte de la Universidad Mayor de San Andrés, se le fue proporcionada una máquina virtual en los servidores centrales, por esa razón el costo es 0.

5.2.6. COSTE TOTAL

El coste total del sistema en USD es la suma de los costos, los cuales se presentan en la siguiente Tabla 5.4:

Tabla 5. 4: Coste total

Descripción	**Costo Total (USD)**
Costo de desarrollo	6000
Coste de elaboración	600
Costo de implementación	0
Total	6600

5.3. ANÁLISIS DE BENEFICIOS

En un proyecto es muy importante analizar la posible rentabilidad de dicho proyecto y sobre todo si es viable o no. Cuando se elaborar un proyecto hay que invertir un capital y se espera obtener una rentabilidad a lo largo de los años. Esta rentabilidad debe ser mayor al menos que una inversión con poco riesgo.

El análisis de los beneficios del proyecto se emplea 5 criterios de evaluación, en el cual es posible comparar y evaluar el nuevo sistema con respecto al anterior.

- Incremento de velocidad
- Capacidad en el volumen de información
- Control de procesos
- Integración de la información
- Información para la toma de decisiones
- Valor neto actual

5.3.1. VALOR ACTUAL NETO

El Valor Actual Neto o (VAN), es un indicador financiero que mide los flujos de los futuros ingresos y egresos que tendrá un proyecto, para determinar si luego de desconectar

la inversión inicial, nos quedara alguna ganancia. Si el resultado es positivo, el proyecto es viable.

De este modo si se tiene varios proyectos sobre la mesa el VAN nos permitirá determinar cuál proyecto es el más rentable entre varias opciones de inversión.

Si VAN > 0 : El proyecto es rentable

Si VAN = 0 : El proyecto es rentable, pero ya está incorporando ganancia de la TD.

Si VAN < 0 : El proyecto no es rentable

La fórmula para calcular el van es la siguiente:

$$VAN = \sum_{t=1}^{n} \frac{Ganancias}{(1+k)^t} - \sum_{t=1}^{n} \frac{Costos}{(1+k)^n}$$

N es el número de periodos considerados.

K la tasa de descuento o tasa de interés al préstamo.

Los Gastos y ganancias que se estiman en un lapso de 4 se muestran en la siguiente tabla 5.5, para el presente caso se utilizará una tasa de descuento del 10%.

Tabla 5. 5: Cálculo del VAN

Año	Costos	Ganancias	Costos/(1+k) ^{n}	Ganancias/(1+k) ^{n}	Resultado
0	6600	0	6600	0	-6600
1	0	3500	0	3181,82	3181,82
2	0	4500	0	3719,01	3719,01
Total	6600	8000	6600	6900,83	300,83

Así,

$$VAN = \sum \frac{Ganancias}{(1+0{,}14)} - \sum \frac{Costos}{(1+0{,}14)}$$

$$VAN = 300{,}83$$

Por lo tanto, podemos afirmar que nuestro proyecto es rentable.

5.3.2. COSTO/BENEFICIO

Para hallar el costo – beneficio de nuestro proyecto aplicamos la siguiente ecuación:

$$Costo\ Beneficio = \frac{\sum Ganancias}{\sum Costos}$$

$$Costo\ Beneficio = 1{,}21$$

Es decir que por cada USD invertido en el proyecto de Software, la institución tendrá una ganancia de 0.21 USD.

5.3.3. TASA INTERNA DE RETORNO

Cuando el VAN toma un valor igual a 0, la variable k pasa a llamarse TIR. Que es la rentabilidad que nos proporcionará el proyecto.

La siguiente ecuación calcula el VAN:

$$VAN = Inversión + \sum \frac{Ganancias}{(1-k)^n}$$

Tenemos:

$$VAN = 6600 + \frac{3500}{1+TIR} + \frac{4500}{(1+TIR)^2}$$

$$(1+TIR)^2 * \ 0 = -6600 * (1+TIR)^2 + 3500 * (1+TIR) \quad + 4500$$

$$1 + TIR = \frac{-3500 \pm \sqrt{3500^2 - 4 * (-6600) * 4500}}{2 * (-6600)}$$

$$1 + TIR = \ -0{,}60 \ y \ 1{,}13$$

Se toma en cuenta el positivo ya que no es posible tener una tasa negativa.

$$TIR = 1{,}13 - 1 = 0{,}13$$

$$TIR = 0{,}13 * 100 = 13\%$$

El tipo de interés ofrecido debe de ser menor o igual al 13%.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

A partir del cumplimiento de los requerimientos establecidos por los Usuarios del Sistema pertenecientes a la de Facultad de Ciencias Puras y Naturales, se ha cumplido con los objetivos planteados por medio de la implementación del sistema de información de autoevaluación en las unidades académicas de pregrado, a través de los diferentes módulos desarrollados para este proyecto.

Por tanto, se llega a las siguientes conclusiones:

- Se finalizó el sistema web de administración para la autoevaluación, que ayuda a simplificar el proceso y mantiene informados a los actores del mismo.

- Se propuso un sistema que toma en cuenta la escalabilidad y posterior integración con otros sistemas, para su comunicación.

- Se generó resultados del proceso de Autoevaluación respecto al marco de referencias del Sistema de la Universidad Boliviana.

- Se generaron reportes finales y medios que proporcionar información para una correcta toma de decisiones de los procesos de autoevaluación.

6.2. RECOMENDACIONES

Para ampliar el presente proyecto de grado, se hacen las siguientes recomendaciones:

- Realizar ampliaciones en cuanto la funcionalidad del Software a más procesos y extenderlos a más áreas de la facultad.

- Aprovechar el uso de las nuevas herramientas de desarrollo aplicadas en este proyecto para futuros desarrollos de Software tanto web como móviles.

- Debido al manejo de Documentación física se recomienda la inclusión de políticas para el uso de documentos digitales en el inicio de los procesos de autoevaluación, como las resoluciones emitidas por las autoridades y así acoplar el registro en el sistema.

BIBLIOGRAFÍA

Hurtado, V., Carvajal, B., Huanca, C.,(2012). Autoevaluación de la Carrera de Informática. Universidad Mayor de San Andrés.

Alamio, M. 2013. Proyectos ágiles con scrum. Argentina: Keleer. 123p

Barrientos. (2002). Guía de Apoyo para los procesos de autoevaluación de carreras de pregrado. Universidad de los Lagos Osorno

http://www.planificacion.umsa.bo/documents/1778193302/0/UMSA+CARRERAS+Y+PROGRAMAS+ACREDITADOS.pdf

Eraso, J. 2013. Aplicación para la gestión de proyectos ágiles con Scrum. La Rioja, Universidad de La Rioja. 51p

Bestbipractices. 2015. Breve resumen sobre SCRUM [en linea] <https://bestbipractices.wordpress.com/2013/07/10/breve-resumen-sobre-scrum/> [consulta: 01 de junio de 2017]

CocomoII. (14 de Abril de 2016). Obtenido de TEMA 7: http://www.eici.ucm.cl/Academicos/ygomez/descargas/Ing_Sw2/apuntes/cocomo_manual_espanol.pdf

Fernandez, Y., & Díaz, Y. 2012. Patrón Modelo-Vista-Controlador. Obtenido de http://revistatelematica.cujae.edu.cu/index.php/tele/article/viewFile/15/10

Nolivos, G., Coronel, F., & Campaña, M. 2010. Iimplementación de un Sistema Web para el Control de un Taller Técnico Automotriz en Plataforma PHP - MySQL utilizando UWE para la Empresa

Palacio, J. 2014. Gestón de Proyectos Scrum Manager. Obtenido de http://www.scrummanager.net/files/sm_proyecto.pdf

Perez-herrera, M. 2015, Arquitecturas basadas en microservicios. Madrid, E.T.S.I. Telecomunicación (UPM). 50p

Pressman, R. 2012. Ingeniería de Software, un enfoque práctico. 7° ed. México, McGraw-Hill Interamericana Editores. 777p

QSM, 2017. Quantitative Software Management. Obtenido de Quantitative Software Management: http://www.qsm.com/resources/function-point-languages-table

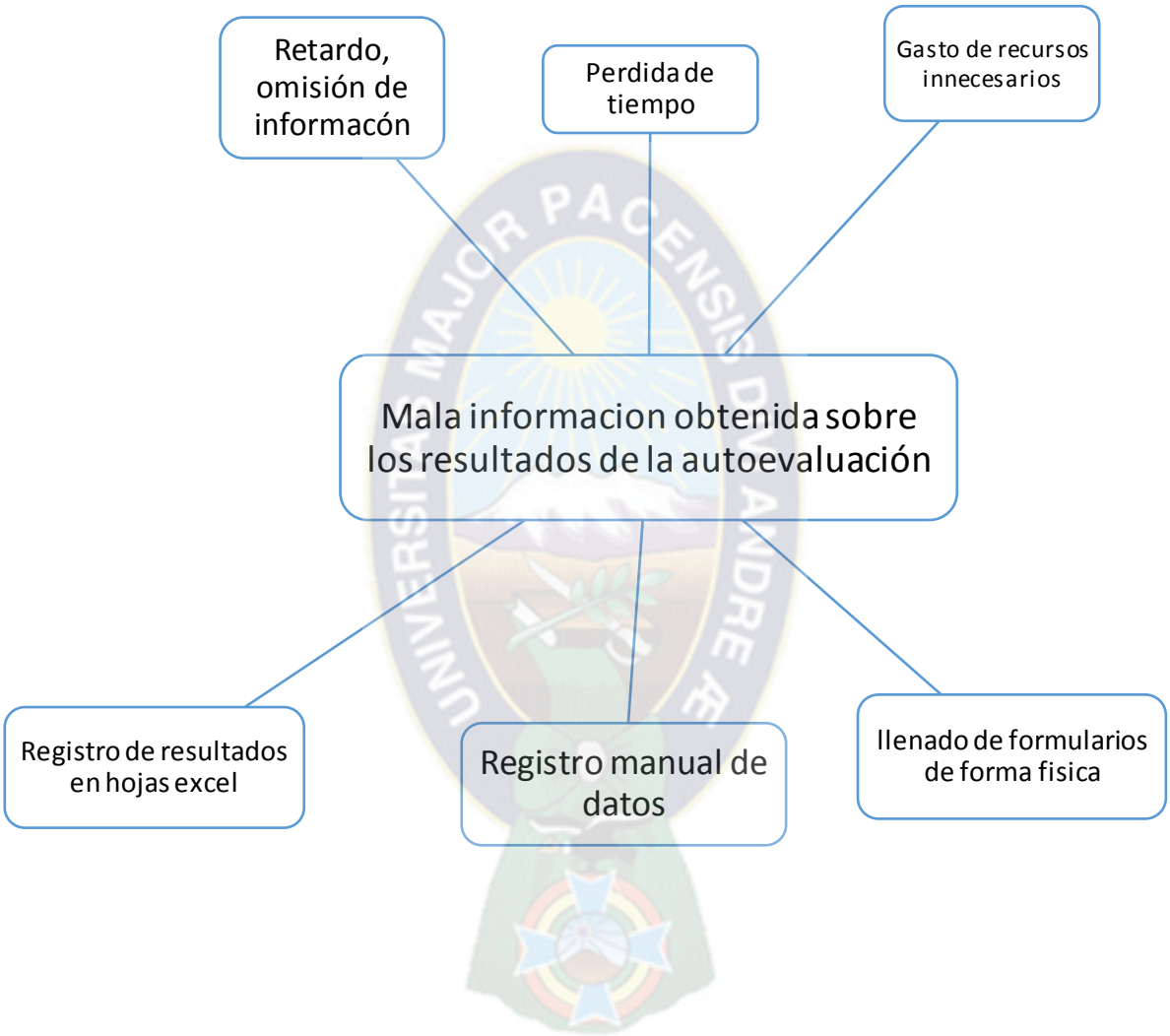
Rodriguez, A. 2009. Metodologías de diseño usadas en la ingeniería web, su vinculación con las ntics. La Plata, Universidad Nacional de La Plata, Facultad de Informática. 58p

(2009). XI Congreso Nacional de Universidades. oruro.

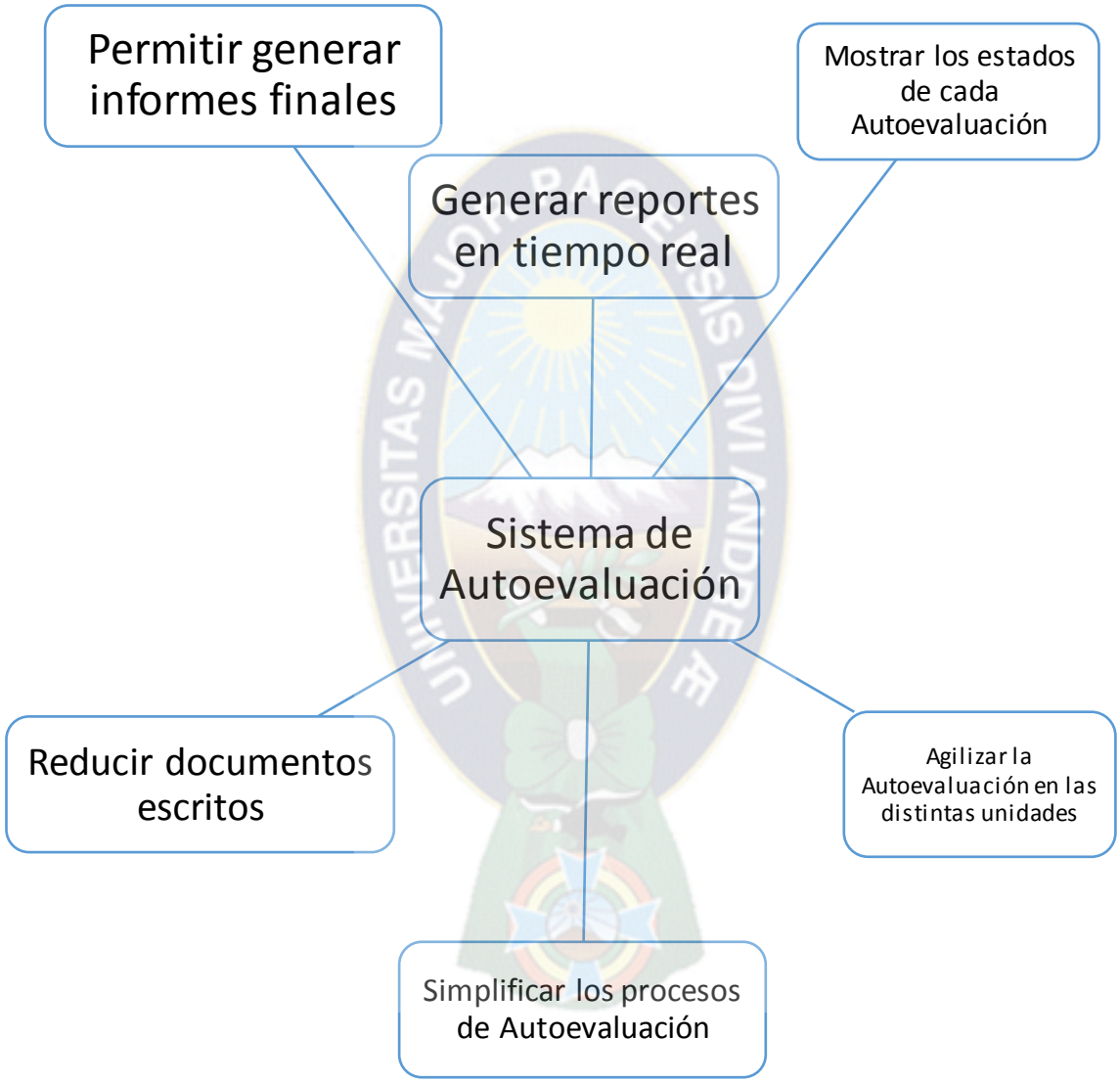
Reglamento de autoevaluacion evaluacion interna y acreditacion de la Universidad Mayor de San Andres, 192/09 (Honorable Consejo Universitario 6 de Mayo de 2009).

ANEXOS

Anexo A -. Árbol de problemas



Anexo B-. Árbol de Objetivos



DOCUMENTOS